FEUP

RAMP FOR CAPACITATED FACILITY LOCATION PROBLEMS

Telmo Manuel Sampaio Pinto de Matos

Department of Industrial Engineering and Management
Faculty of Engineering, UNIVERSITY OF PORTO

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Industrial Engineering and Management.

Supervisor: José Fernando da Costa Oliveira
FEUP - Department of Industrial Engineering and Management
Co-Supervisor: Dorabela Regina Chiote Ferreira Gamboa
Porto Polytechnic – School of Management and Technology

Faculdade de Engenharia da Universidade do Porto
2017

Resumo

Problemas de Localização de Instalações são problemas complexos de otimização combinatória cujo objetivo é determinar um conjunto de localizações onde colocar instalações, de forma a satisfazer a procura de um determinado número de clientes com custo mínimo. Dependendo do tipo de problema, diferentes tipos de restrições podem ser definidos, tais como, capacidades limitadas que as instalações podem possuir, os clientes poderem ser servidos por uma única ou por várias instalações, ou outras restrições.

Tratando-se de problemas NP-difíceis, a utilização de métodos exatos para resolver instâncias grandes destes problemas pode ser seriamente comprometida pelos elevados tempos computacionais necessários para obter a solução ótima. Assim, torna-se necessário desenvolver métodos heurísticos eficazes para a resolução deste tipo de problemas.

RAMP é uma abordagem metaheurística que explora o lado primal e dual de um problema e seu relacionamento, orientando a pesquisa com base em princípios de memória adaptativa. Este método já provou o seu potencial através da obtenção de soluções de grande qualidade para vários problemas complexos de otimização combinatória, tais como o problema da Atribuição Generalizada, o problema de Ordenação Linear, o problema de Localização de Instalações sem Restrições de Capacidade, entre outros. Neste contexto, o objetivo principal deste trabalho é o desenvolvimento de aplicações de RAMP para outras variantes do problema de Localização de Instalações com restrições de Capacidade, motivados por aplicações do mundo real, no domínio das telecomunicações, do abastecimento de água e dos transportes, entre outras. O sucesso obtido com a aplicação RAMP para o problema de Localização de Instalações sem Restrições de Capacidade, sugeria que o uso da abordagem RAMP para outros problemas de Localização também poderá produzir resultados de elevada qualidade. Ficou demonstrado neste trabalho que o uso desta metaheurística é eficaz na resolução de três problemas distintos: o problema de Localização de Instalações com Restrições de Capacidade, o problema de

Localização de Hubs com Afetação Simples e com Restrições de Capacidade e o problema de Localização de p-Hubs com Afetação Simples e com Restrições de Capacidade.

A primeira parte deste trabalho contém uma introdução, onde é introduzida a motivação que deu origem a este trabalho e uma explicação dos problemas que serão estudados. De seguida será apresentado o estado da arte para esses mesmos problemas seguido da explicação do método proposto para a resolução dos problemas. Por fim, serão abordados individualmente os problemas escolhidos e quais as técnicas utilizadas na sua resolução.

A segunda parte desta tese reflete os artigos que envolvem cada um dos três problemas propostos e que foram escritos ao longo do Programa Doutoral. Em cada um destes artigos é possível analisar em detalhe as diferentes estratégias propostas para a resolução destes problemas difíceis com diferentes características e suscetíveis a diferentes abordagens.

Palavras-chave: Facility Location Problems, Hub Location Problems, Metaheuristics, Relaxation Adaptive Memory Programming, Combinatorial Optimization.

Abstract

Facility Location Problems are complex combinatorial optimization problems that aim to locate a set of facilities to serve a specific set of customers with minimum cost. Depending on the real-world application, different types of constraints may be defined, such as, limited capacities of the facilities, the clients may be served by only one or more facilities, among others. Being NP-Hard problems, using exact methods to solve large instances of these problems can be seriously compromised by the high computational times required to obtain the optimal solution. Thus, it is necessary to develop efficient heuristic methods to solve such problems.

RAMP is a new Primal-Dual metaheuristic framework that explores the primal and dual side of a problem, and their relationship, together with adaptive memory principles. This framework has already proved its potential by obtaining high quality solutions for several complex combinatorial optimization problems, such as the Generalized Assignment Problem, the Linear Ordering Problem or the Uncapacitated Facility Location Problem, among others. In this context, our primary focus is to apply RAMP to other complex Capacitated Facility Location Problems, which are central in real-world application in the telecommunications, water supply, and transportation sector. The success obtained with the application of the RAMP framework to the Uncapacitated Facility Location Problem, suggested that the use of the RAMP approach to solve other Location Problems will also produce state of the art algorithms. It was confirmed in this study that the use of this metaheuristic proved to be effective on solving three distinct problems: the Capacitated Facility Location Problem, the Capacitated Single Allocation Hub Location Problem and the Capacitated Single Allocation p-Hub Location Problem.

The first part of this work contains an introduction, where the motivation that led to this work is presented, together with an explanation of the problems to be studied. Afterwards a state of the art for these same problems is presented, followed by the explanation of the proposed method for solving the

problems. Finally, these problems will be individually addressed and the techniques used in its resolution explained.

The second part of this thesis reflects the papers tackling each one of the three problems, and that have been written during the PhD. In each of these papers it is possible to analyze in detail the strategies used to solve each problem, presenting different characteristics and therefore susceptible to different approaches.

Keywords: Facility Location Problems, Hub Location Problems, Metaheuristics, Relaxation Adaptive Memory Programming, Combinatorial Optimization.

Preface

This thesis has been submitted at the Department of Industrial Engineering and Management, Faculty of Engineering, University of Porto in partial fulfilment of the requirements for acquiring the degree of Doctor of Philosophy in Industrial Engineering and Management.

The work was supervised by Professor José Fernando Oliveira from the Department of Industrial Engineering and Management, Faculty of Engineering, University of Porto and co-supervised by Professor Dorabela Gamboa, from the Polytechnic of Porto.

This work addresses three combinatorial optimization problems and consists of two parts. The first part describes the problems, reviews the literature and presents the framework used to solve them. The second part of the thesis contains three scientific papers that reflect the research done during the time of this study.

Acknowledgment

The author wishes to express sincere gratefulness to Professor Dorabela Gamboa, from the Polytechnic of Porto - School of Management and Technology for patience and dedication in reviewing, commenting and direct monitoring in the execution of all the work presented here. To Professor José Fernando Oliveira from the Department of Industrial Engineering and Management, Faculty of Engineering, University of Porto for collaboration and also reviewing of the thesis.

To all my co-workers of the Polytechnic of Porto - School of Management and Technology, I would also like to express my thanks. They have been very helpful in the implementation of this thesis.

To all my family (both families – you will be always parents, brothers and sisters) and friends, that I care very much, and to my life mate, for patience and support.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

RAMP	Relaxation Adaptive Memory Programming
PD-RAMP	Primal-Dual RAMP
FLP	Facility Location Problems
UFLP	Uncapacitated Facility Location Problem
CFLP	Capacitated Facility Location Problem
HLP	Hub Location Problems
CSAHLP	Capacitated Single Allocation Hub Location Problem
CSApHLP	Capacitated Single Allocation p-Hub Location Problem
SCR	Surrogate Constraints Relaxation
LR	Lagrangean Relaxation
TS	Tabu Search

RAMP for Capacitated Facility Location Problems

1. Introduction

This thesis addresses the Capacitated Facility Location Problem (CFLP) and two extensions of the Hub Location Problem (HLP): the Capacitated Single Allocation Hub Location Problem (CSAHLP) and the Capacitated Single Allocation p-Hub Location Problem (CSApHLP). These have raised a lot of interest in the literature and have a huge number of practical applications, going from the telecommunications to the transport sectors and are considered very difficult to solve. Obtaining an optimal solution for Facility Location or hub Location problems does not always pay off when this solution involves large computational times. Thus, it becomes necessary to design algorithms able to solve these problems in shorter times, which is not compatible with the resolution of these problems by exact methods. To tackle these problems, a metaheuristic called RAMP (Relaxation Adaptive Memory Programming) was used, given its previous success involving the resolution of other combinatorial optimization problems.

This work was carried out over the PhD and the goal was the application of the RAMP framework to CFLP, CSAHLP and then to CSApHLP. The success obtained with the application of the proposed framework to CFLP led to build an algorithm to solve the CSAHLP based on the same framework. The excellent results obtained with RAMP for CSAHLP suggested that the application to another version of Hub Location Problem, the CSApHLP, could also produce state of the art results. It was confirmed by the excellent results obtained with the RAMP framework.

The work presented in this thesis is divided into two parts. The first part, including the introduction, describes the problems where RAMP was applied, and reviews the literature for all the problems here tackled. The second part

of the thesis contains three scientific papers reporting the research concluded during the time of this PhD program. The following subchapter introduces the background and motivation for the research project, details its goals and objectives and provides an outline of each chapter of the thesis.

1.1. Background and Motivation

The research thesis has built on recent research work and aims to investigate the application of Relaxation Adaptive Memory Programming (RAMP) approach to the Capacitated Facility Location Problem (an FLP extension), the Capacitated Single Allocation Hub Location Problem and the Capacitated Single Allocation p-Hub Location Problem (both HLP extensions), for which current state of the art algorithms struggle to find high quality solutions.

The FLP main objective (a very general definition by Drezner [33]) is to "Determine the location of one or more facilities in a way that optimizes certain objectives such as minimizing transportation costs, providing equitable service to customers, or minimizing the time taken to deliver emergency services." This definition is very generic, and depending on the characteristics of the FLP variant we can have more specific definitions. This type of problems has many real-world applications, such as, choosing the best location to open a hospital or a school, defining the best mail delivery system of a country, among others. One of the well-known extensions of the FLP family is the Capacitated Facility Location Problem (CFLP) that belongs to the class of hard combinatorial optimization problems. The CFLP general goal is to locate a set of facilities with limited capacity that will serve the demand of a particular set of customers with minimum cost.

Concerning the HLP, and introducing the hub concept that can be defined as "central facilities which act as switching points in networks connecting a set of interacting nodes" (see O'Kelly [92]), the objective is to determine which nodes are to be used as hubs and the allocation of the other nodes (or spokes) to the hubs, in order to route the flows between the origin and destination pairs. As for FLP, this is a generic definition and depends on the characteristics of each HLP extension, such as, single-allocation or multiple-allocation, number of hubs to be opened is fixed or not, etc. One of the well-known extensions of

the HLP is the Capacitated Single Allocation Hub Location Problem (CSAHLP) and the Capacitated Single Allocation p-Hub Location Problem (CSApHLP) that belongs to the class of hard combinatorial optimization problems. The objective of the CSAHLP is to locate the hubs with limited capacity and to allocate the nodes to a single hub so that the total transportation cost is minimized. The CSApHLP is very similar and the only difference from the previous problem is that the number of nodes to be set as hubs is known.

The recent success of RAMP applications to a wide variety of hard combinatorial optimization problems, suggests that the application of the RAMP framework to FLP and HLP will produce competitive results, challenging the best-known algorithms for the resolution of these problems.

RAMP is a primal-dual metaheuristic framework presented by César Rego [100] in 2005, which explores the primal and dual side of a problem and their relationship. This framework has already proved its potential by obtaining high quality solutions for several complex combinatorial optimization problems, such as the Minimum Spanning Tree Problem [101], Multi-Resource Generalized Assignment Problem [102], Resource Constraint Project Scheduling [106] or the Uncapacitated Facility Location Problem [45], among others. In this context, our primary focus is to develop RAMP applications for other complex Facility and Hub Location Problems, embracing different strategies for solving them. RAMP's unique combination of metaheuristic and mathematical programming techniques presents new opportunities to exploit new approaches for solving these difficult combinatorial optimization problems leading to better solutions, both in terms of time and quality.

Based on this approach we intend to design, implement and test new RAMP algorithms capable of efficiently handling both Hub and Facility Location Problems and verify the effectiveness of this promising metaheuristic.

1.2. Problem Presentation

This document addresses three combinatorial optimization problems: the well-known Capacitated Facility Location Problem (an FLP extension), the Capacitated Single Allocation Hub Location Problem and the Capacitated Single

Allocation p-Hub Location Problem (both HLP extensions). Bellow will be presented a possible formulation for each one of these problems.

Capacitated Facility Location Problem (CFLP)

The Capacitated Facility Location Problem (CFLP) is a well-known combinatorial optimization problem belonging to the class of the NP-Hard problems [46]. The CFLP can be formulated as follows:

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n} D_j C_{ij} x_{ij} + \sum_{i=1}^{m} F_i y_i \quad (1)$$

$s.t.$

$$\sum_{i=1}^{m} x_{ij} = 1, \;\; j = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i y_i, \;\; i = 1, \dots, m \quad (3)$$

$$x_{ij} \geq 0 \,, \;\; j = 1, \dots, n \;\; i = 1, \dots, m \quad (4)$$

$$y_i \in \{0,1\}, \;\; i = 1, \dots, m \quad (5)$$

where m represents the number of possible locations to open a facility and n the number of customers to be served. S_i indicates the capacity of facility i and F_i the fixed cost for opening that facility. D_j represents the demand of client j and C_{ij} the unit shipment cost between facility i and customer j. The variable x_{ij} denotes the amount shipped from facility i to costumer j and y_i indicates whether facility i is open or not. The objective is to locate a set of facilities in such way that the sum of the costs for opening those facilities and the transportation costs for serving all customers is minimized.

Given a set of open facilities ($y_i = 1, \; i \in I$), the CFLP has the particularity of becoming a Transportation Problem (TP) that can be solved in polynomial time. TP can be formulated as:

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n} D_j C_{ij} x_{ij} \quad (6)$$

s.t.

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i, \quad i = 1, \dots, m \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \dots, n \tag{8}$$

$$x_{ij} \geq 0\,, \quad j = 1, \dots, n \quad i = 1, \dots, m \tag{9}$$

Where C_{ij} is the unit shipment cost from costumer j to facility i, x_{ij} is the amount sent from costumer j to facility i, S_i is the availability of facility i and D_j is the demand of costumer j. The objective is determining an optimal transportation scheme between the facilities and customers so that the transportation costs are minimized. Also, if we eliminate the capacity of each facility (eliminating in equation (3) variable S_i) and make $D_j = 1$ in equation (2) we get the Uncapacitated variant of the problem (UFLP), also widely studied in the literature.

Capacitated Single Allocation Hub Location Problem

The Capacitated Single Allocation Hub Location Problem (CSAHLP) is a well-known combinatorial optimization problem belonging to the class of the NP-Hard problems [93]. The model used in this work was proposed by Contreras *et al*. [23] and is described as follows. Consider the complete graph $G = (N, A)$, where N is the set of nodes $N = \{1, 2, \dots, n\}$, that correspond to origin/destinations as well as potential hub locations. Let w_{ij} be the flow between i and j and $O_i = \sum_{j \in N} w_{ij}$ is the outgoing flows from node $i \in I$ and $j \in N$ and $D = \sum_{i \in N} O_i$ is the total flow generated in the graph. For each node i, b_i $(i \in N)$ denote the capacity and f_i $(i \in N)$ the fixed set-up cost of hub i. The capacity of a hub represents an upper bound on the total incoming flow that can be processed in the hub. Thus, it refers to the sum of the flow generated at the nodes that are assigned to the hub. The distance between nodes i and j is denoted as d_{ij}. We will use these distances as a measure of the per unit flow transportation costs along the links of the graph. These distances are weighted

by some discount factor, denoted χ, α and δ, to represent the collection, transfer and distribution costs per unit of flow, respectively. More precisely:

- The Collection cost is the cost incurred on flow from a given node to a hub (cost of the node-to-hub).
- The Transfer cost is the cost of the flow between hubs (cost of the hub-to-hub flow).
- The Distribution cost denotes the cost of the flow from a hub to the node (cost of hub-to-node flow).

The objective consists of choosing the set of nodes to be established as hubs and minimize the total cost of the allocation of all non-hubs to the chosen hubs, without violating the capacity constraint of the hubs.

The cost of routing one unit of flow along the path $i-j-k-m$ (these are the paths between origin destination pairs, where i and j represent the origin and destination, respectively, and k and m are the hubs to which i and j are allocated, respectively) is given by:

$$\mathrm{F_{ijkm}} = \mathrm{w_{ij}}(\chi \mathrm{d_{ik}} + \alpha \mathrm{d_{km}} + \delta \mathrm{d_{mj}}) \tag{10}$$

and for each pair i, $k \in N$ the following sets of binary decision variables are defined by:

$$\mathrm{Z_{ik}} = \begin{cases} 1 \text{ if node i is assigned to hub k;} \\ 0 \text{ otherwise.} \end{cases} \tag{11}$$

When $i = k$, variable Z_{kk} represents the establishment or not of the hub at node k. The Z variables will be referred to as location/allocation variables. We define an additional set of binary variables that represent the existence of flow through each link of the graph. For each $i,j,k,m \in N$, the existence of flow through each link of the graph is defined by the following set of binary variables

$$\mathrm{X_{ijkm}} = \begin{cases} 1 \text{ if flow from i to j goes via hubs k and m;} \\ 0 \text{ otherwise.} \end{cases} \tag{12}$$

The mathematical formulation for CSAHLP is:

$$\text{Min} \sum_{k\in N} f_k Z_{kk} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N} F_{ijkm}\, X_{ijkm} \tag{13}$$

s.t.

$$\sum_{k\in N}\sum_{m\in N} X_{ijkm} = 1 \;\forall\, i,j \in N \tag{14}$$

$$Z_{ik} \le Z_{kk} \;\forall\, i,k \in N \tag{15}$$

$$\sum_{m\in N} X_{ijkm} = Z_{ik} \;\forall\, i,j,k \in N \tag{16}$$

$$\sum_{k\in N} X_{ijkm} = Z_{jm} \;\forall\, i,j,m \in N \tag{17}$$

$$\sum_{i\in N} O_i z_{ik} \le b_k Z_{kk} \;\forall\, k \in N \tag{18}$$

$$\sum_{k\in N} b_k Z_{kk} \ge D \tag{19}$$

$$Z_{ik} \in \{0,1\} \;\forall\, i,k \in N \tag{20}$$

$$X_{ijkm} \in \{0,1\} \;\forall\, i,j,k,m \in N \tag{21}$$

Constraints (14) guarantee that every node is assigned to one single hub, whereas constraints (15) impose that a non-hub node is assigned to a hub node. Constraints (16) state that if node i is assigned to hub k then all the flow from node i to any other (fixed) node j must go through some other hub m. Constraints (17) have a similar interpretation relative to the flow arriving to a node j assigned to hub m from some node i. Constraints (18) ensure that the overall incoming flow of nodes assigned to a hub does not exceed its capacity. Constraint (19) is the aggregated demand constraint. Note that satisfying the aggregated demand constraint is a necessary condition for location/allocation vectors being feasible, that can be derived by adding up all constraints (18), and considering equalities (14) and (16). Constraint (19) is redundant for the formulation, as it refers in the work of Contreras [23], but the author chooses to include in the model in order to strengthen the relaxation. We include it also

in order to apply Lagrangean Relaxation, as we will be able to see later in this document.

Capacitated Single Allocation p-Hub Location Problem

The Capacitated Single Allocation p-Hub Location Problem (CSApHLP) is a well-known combinatorial optimization problem belonging to the class of NP-Hard problems [94]. The model used in this work was proposed by Contreras *et al*. [23] (considering the fixed set of required hubs) and is described as follows.

Consider the complete graph $G = (N, A)$, where N is the set of nodes $N = \{1, 2, \dots, n\}$, that correspond to origin/destinations as well as potential hub locations. Let w_{ij} be the flow between i and j and consider that $O_i = \sum_{j \in N} w_{ij}$ the outgoing flows from node $i \in I$ and $j \in N$ and $D = \sum_{i \in N} O_i$ is the total flow generated in the graph. For each node i, b_i $(i \in N)$ denote the capacity and f_i $(i \in N)$ the fixed set-up cost of hub i. The capacity of a hub represents an upper bound on the total incoming flow that can be processed in the hub. Thus, it refers to the sum of the flow generated at the nodes that are assigned to the hub. The distance between nodes i and j is denoted as d_{ij}. We will use these distances as a measure of the per unit flow transportation costs along the links of the graph. These distances are weighted by some discount factor, denoted χ, α and δ, to represent the collection, transfer and distribution costs per unit of flow, respectively (see the description of CSAHLP for complete explanation about these discount factors).

The objective is to minimize the total cost of the allocation of all non-hubs to the set of p nodes chosen to be hubs while not violating the capacity constraint of the hubs. The cost of routing one unit of flow along the path $i - j - k - m$ (these are the paths between origin destination pairs, where i and j represent the origin and destination, respectively, and k and m are the hubs to which i and j are allocated, respectively) is given by:

$$\mathrm{F_{ijkm}} = \mathrm{w_{ij}}(\chi \mathrm{d_{ik}} + \alpha \mathrm{d_{km}} + \delta \mathrm{d_{mj}}) \tag{22}$$

For each pair $i, k \in N$ the following sets of binary decision variables are defined by:

$$Z_{ik} = \begin{cases} 1 \text{ if node i is assigned to hub k;} \\ 0 \text{ otherwise.} \end{cases} \quad (23)$$

When $i = k$, variable Z_{kk} represents the establishment or not of a hub at node k. The Z variables will be referred to as location/allocation variables. We define an additional set of binary variables that represent the existence of flow through each link of the graph. For each $i, j, k, m \in N$ let the amount of flow through each link of the graph is defined by the following set of binary variables

$$X_{ijkm} = \begin{cases} 1 \text{ if flow from i to j goes via hubs k and m;} \\ 0 \text{ otherwise.} \end{cases} \quad (24)$$

The mathematical formulation for CSAHLP is:

$$\text{Min} \sum_{k \in N} f_k Z_{kk} + \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{m \in N} F_{ijkm} X_{ijkm} \quad (25)$$

s.t.

$$\sum_{k \in N} \sum_{m \in N} X_{ijkm} = 1 \; \forall \, i, j \in N \quad (26)$$

$$\sum_{k \in N} Z_{kk} = p \quad (27)$$

$$Z_{ik} \leq Z_{kk} \; \forall \, i, k \in N \quad (28)$$

$$\sum_{m \in N} X_{ijkm} = Z_{ik} \; \forall \, i, j, k \in N \quad (29)$$

$$\sum_{k \in N} X_{ijkm} = Z_{jm} \; \forall \, i, j, m \in N \quad (30)$$

$$\sum_{i \in N} O_i z_{ik} \leq b_k Z_{kk} \; \forall \, k \in N \quad (31)$$

$$\sum_{k \in N} b_k Z_{kk} \geq D \quad (32)$$

$$Z_{ik} \in \{0,1\} \; \forall \, i, k \in N \quad (33)$$

$$X_{ijkm} \in \{0,1\} \; \forall \, i, j, k, m \in N \quad (34)$$

Constraints (26) guarantee that every node is assigned to one single hub. Constraint (27) specifies that exactly *p* nodes will act as hubs, whereas constraints (28) impose that a non-hub node is assigned to a hub node. Constraints (29) state that if node i is assigned to hub k then all the flow from node i to any other (fixed) node j must go through some other hub m. Constraints (30) have a similar interpretation relative to the flow arriving to a node j assigned to hub m from some node i. Constraints (31) ensure that the overall incoming flow of nodes assigned to a hub does not exceed its capacity. Constraint (32) is the aggregated demand constraint. Note that satisfying the aggregated demand constraint is a necessary condition for location/allocation vectors being feasible, that can be derived by adding up all constraints (31), and considering equalities (26) and (29). Constraint (32) is redundant for the formulation, as it refers in the work of Contreras *et al.* [23], but the author chooses to include in the model in order to strengthen the relaxation. We include it also in order to apply Lagrangean Relaxation, as we will see later in this work.

1.3. Thesis Structure

This thesis is structured in two separate parts. The first part contains an introduction to the problems addressed in this thesis as well as the resolution strategies explored in this PhD study. Aside from this introductory chapter, the first part consists of four separate chapters:

- Chapter 2 describes the main characteristics of the problems to be addressed, reviews the literature and discusses in further detail the most commonly used solution methods for these problems.
- Chapter 3 presents the framework that was used to solve the proposed problems. It will be explained the application of this solution procedure to other optimization problems.
- Chapter 4 lists the main contributions of the work conducted during this PhD study. This includes a detailed overview of the work contained in the research papers that constitutes Part II of this thesis.

- Finally, Chapter 5 highlights the contributions of this thesis and reflects on possible directions for future research within the application of RAMP to other combinatorial optimization problems.

The second part of this thesis contains the three research papers written during the PhD study:

- Chapter 6 addresses the Relaxation Adaptive Memory Programming (RAMP) approach to the Capacitated Facility Location Problem. This method combines Lagrangean Relaxation with Subgradient Optimization with Tabu Search to explore primal-dual relationships as a way to create advanced memory structures that integrate information from both primal and dual solution spaces. The algorithm was tested on the standard ORLIB dataset and on other very large-scale testbed for the CFLP and has efficiently found the optimal solution for all instances from ORLIB and very competitive results for the larger ones. Comparisons with the current best performing algorithms for the CFLP highlights our RAMP algorithm excellent results.
- Chapter 7 addresses the Capacitated Single Allocation Hub Location Problem. We propose a Relaxation Adaptive Memory Programming approach for the CSAHLP. Our method combines Lagrangean Relaxation with Subgradient Optimization in dual side with an improvement method to explore the primal side. The algorithm was tested on the standard dataset and produced extremely competitive results with the most recent best solutions known for this problem.
- Chapter 8 addresses the Capacitated Single Allocation p-Hub Location Problem. The algorithm for solving this problem combines Lagrangean Relaxation with Subgradient Optimization with a very simple local search algorithm together with an improvement method to explore primal-dual interconnection. Although the simplest version of the algorithm produced extremely competitive results, we decide to implement a more sophisticated RAMP algorithm for the resolution of this problem. Both algorithms were tested on the standard dataset and compared with current best performing algorithms in literature. The results obtained for

this problem outperformed the best results in the literature demonstrating the excellent performance of the RAMP framework.

2. State of the art

In this chapter, we address the Facility Location problem, the Hub Location problem and its extensions. They have an important presence in the Operations Research literature since the early 60's dealing with real situations such as where to place warehouses, schools, fire stations or hospitals, in what concerns the FLP or dealing with postal delivery services, emergency care services, network supply in supermarket chains, location of antennas for communication, among many others, in what the HLP is concerned. Below we will see in detail the most recent approaches for each one of the above-mentioned location problems.

Facility Location Problems

Facility Location Problems (FLP) are widely studied in the literature with several practical applications, reaching areas such as telecommunications, design of a supply chain management, transport utilities and water distribution networks. They belong to the NP-Hard class of problems [90] and are considered extremely difficult to solve. The definition for FLP, depends on its variant but generically, the problem always refers to determining locations to open facilities. A possible definition for FLP can be to "determine the location of one or more facilities in a way that optimizes certain objectives such as minimizing transportation costs, providing equitable service to customers, or minimizing the time taken to deliver emergency services"[33].

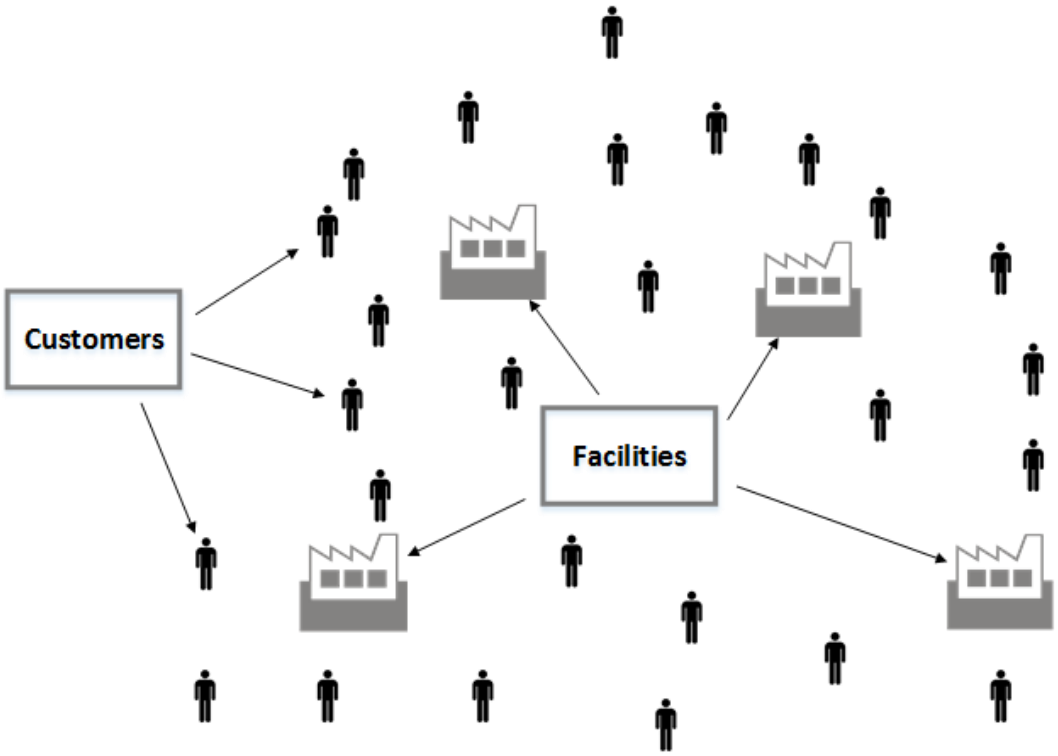

Figure 1: Facility Location problem example.

Figure 1 allows the visualization of the problem of locating facilities satisfying a set of customers, where the objective is, based on distances, to link customers to facilities in such way that the overall distance is minimized. For more information about Facility Location models the reader is referred to the work of [29,34,40,104].

Depending on the FLP constraints, several problem variants may be defined addressing different questions: a facility can have a limited capacity, a client can be satisfied by one facility or more, the facilities can be connected, among others. Depending on the Discrete Facility Location Problem under analysis, different formulation categories may be considered: Minisum Problems (median), Minimax Problems (center) and Covering Problems.

The Minisum problems take in consideration the minimization of the total costs/distances between the facilities and their clients. Examples of this category of problems are the Uncapacitated and Capacitated problem, the single and multiple facility problems, and the p-median problems. A real-world application is the allocation problem between warehouses and stores, where an organization must define the best locations for its warehouses with the goal

of minimizing the warehouses operational costs and the transportation cost between the warehouses and its clients.

The minimax facility location problem goal is to minimize the longest facility/client distance. While Minisum goal is the minimization of the sum of transportation costs between some facilities and clients, the goal for minimax is the minimization of the longest distance, travel time or any other cost function. An example in this category is the p-center problem. As a practical example, an emergency service must arrive at its destinations in the least time possible, so this problem tries to minimize the emergency station radius of operations/action. In this example, the transportation cost is not the primary concern.

The Covering problem goal is to maximize the number of clients covered by a facility within some distance or time. An example of this category is the Maximal Covering Location Problem, where the objective is to cover as many clients as possible. As a practical example in a distributed service, the objective is to locate a facility that can serve the highest number of clients, minimizing the service time. Distance is not in the objective function, but it is a constraint of the problem. Based on these three main formulations, other formulations can be derived to solve different problems. More problem formulations about Covering problems are presented in [33,34,40,124].

The Hub Allocation Problem is a good example, where some clients' demands pass through several facilities until they reach their final destination. An example is the mail delivery system where mail from a region is collected and them redistributed to other mail hubs until it reaches the pretended destination. Another problem formulation is the Undesirable Facility Location Problem, where an undesirable facility (like a waste disposal or an airport) must be located as far away as possible from the population, but close enough to satisfy their needs. For additional problem formulations refer to Foundations of Location Analysis [124] or Facility Location: Applications and Theory [34,124].

In the next subchapter, a well-known FLP extension, which has been very much discussed in the scientific community, will be introduced.

2.1.1. Capacitated Facility Location Problem

The FLP with limited capacity is widely studied in the literature. The classic problem (is one of the basic problems in location theory) is known as Capacitated Facility Location Problem (CFLP), or Capacitated Plant Location Problem or even Capacitated Warehouse Location Problem, and has many extensions and variations. The classic CFLP consists on minimizing the total cost of opening facilities (fixed costs) plus the cost of serving all customers (transportation costs). Furthermore, each facility is not allowed to supply more commodities/goods than its capacity and the demand of each customer has to be met. In practical application, the CFLP has different constraints that affect and complicate the basic problem. This complexity is due to the very different characteristics of the problem's many application areas and economy sectors. Therefore, the CFLP problem has been extensively studied over the past 50 years, resulting in a large number of algorithmic approaches based on exact and heuristic methods. A summary of some of the latest papers and surveys is presented in Table 1.

Heuristic/Lagrangean Relaxation thecnique	[76] Local Search/mathematical programing (2016) [98] FireFly/Genetic Algorithm (2014) [60] Kernel Search (2012) [17] Bee Algorithm (2012) [120] Tabu Search (2011) [78] Benders Decomposition and Genetic Algorithm (2010) [6] Lagrangean Relaxation with Branch-and-Cut (2008) [22] Aproximation algorithm (2004) [14] ADD/DROP (2003) [9] Lagrangean Relaxation (1983)
Exact	[78] Benders Decomposition and Genetic Algorithm (2010) [5] Branch-and-Cut-and-Price (2009) [6] Lagrangean Relaxation with branch-and-cut (2008) [1],[109],[31] Branch & Bound (1977),(1969),(1969) [66] Facility Location and Supply Chain Management – A comprehensive review (2007)
Surveys	[68] A modeling framework for facility location of medical services for large-scale emergencies (2007) [105] Location analysis: A synthesis and survey (2005) [71] Facility location models for distribution system design (2005) [29] Facility location in supply chain design (2005) [33] Facility location: a survey of applications and methods (1995)

Table 1: Summary of the latest papers and surveys on CFLP.

What is, probably, the first heuristic for the CFLP was proposed by Jacobsen [67]. This heuristic was mainly based on moving from one solution to another with simple movements. These methods are very simple and yet effective. The ADD method (based on Kuehn e Hamburger [74] heuristic for the UFLP), starts with all facilities closed and then opens the facilities one by one, led by the best gain. The DROP procedure (based on Feldman *et al.* [42] heuristic for the UFLP) is a greedy heuristic and it is the opposite of the ADD procedure. In each iteration, a facility is dropped from the set of open facilities at the location where the largest saving is obtained.

Relaxation techniques are frequently used to solve the CFLP effectively. Cornuejols *et al*. [25] proposed several algorithms based on relaxations to solve large instances of this problem. Guignard e Spielberg [63] presented a Dual Ascent procedure for CFLP following an idea initially proposed by Bilde and Krarup [13] and Erlenkotter [35] for solving the UFLP. Ling [125], Lorena and Senne [82] and Beasley [9] derived good upper and lower bonds based on relaxation technique, originating good results for the CFLP. Sridharan [112] proposed the use of cross decomposition, following by Van Roy [108], in which the main idea is the exploration between the primal and the dual sides of the problem with the use of subproblems. Bornstein [14] proposed an ADD/DROP procedure, which extends the already mentioned ADD procedure and uses the DROP method to close facilities, if this improves the objective function.

Exact algorithms have also been used to solve the CFLP, based on Branch & Bound [72]. Branch & Bound is a divide and conquer procedure and can be seen as the solution of n sub problems SP_n of the original problem P, which are provenly easier to solve. This procedure takes advantage of good upper and lower limits for the n sub problems, and removes the other sub problems that are not within these limits. At the end, the sub problems and its solutions are gathered to obtain the original problem P. Avella and Boccia [5] presented an exact algorithm based on a Branch-and-Cut-and-Price for solving the CFLP where they solve very large instances up to 1000 facilities and 1000 clients.

Ming-Che Lai *et al*. [78] applied the Benders Decomposition, initially proposed by Benders [12], and use Genetic Algorithms [111] (GA) for the main problem. Once again, the main idea is to divide the original problem in sub-problems more easily to solve. The original problem is relaxed and the constraints and variables that compose the relaxed problem are divided and solved separable, alternating between the divided problem and the original problem. The main idea of GA is to try to produce the natural selection and species evolution belonging to the evolutionary algorithms. It generates elite solutions (populations) and then a local search, selection and mutation factors are applied to generate new populations trying to achieve the optimal solution.

Tabu Search (TS) [58] has many applications in optimization problems with high quality results. More recently, Minghe Sun [120], proposed a TS for

the CFLP with state of the art results. Sun uses memory structures to search in space regions that can be promising. These space regions are kept in long term memory to be intensively explored. TS uses diversification and intensification strategies and the neighborhood based on changing the status of facilities (open-close or close-open).

A new and recent algorithm was proposed by Guastaroba and Speranza [60] to solve the CFLP. This algorithm (kernel search) is a heuristic framework based on the idea of identifying interesting subsets of variables and in solving a sequence of MILP (mixed integer linear problem) problems, each of them restrained to one of the identified subsets of associated variables. For more information about kernel search and its applications can be obtained from [60] and [61].

Other recent metaheuristics based on nature observation are the Firefly-Genetic Algorithm proposed by Rahmani and Mirhassani [98], which is motivated by the social behavior of fireflies and the phenomenon of bioluminescent communication, the Ant Colony Optimization proposed by Venerales and Moscardini [123] and the Bee Colony hybridization proposed by Levanova and Tkachuk [81] with Mixed Integer Programming, which has also produced good results for the CFLP in acceptable computation times.

Hub Location Problems

Hub Location Problems (HLP) have been extensively studied in the literature under several practical applications, being the mail delivery systems the most common example used to explain these problems[1]. HLP belongs to the NP-Hard class of problems [94] and are considered very hard to solve. The HLP goal is to decide which nodes should be used as hubs and to determine the allocation of the other nodes (also called spokes) to the hubs, in order to route the flows between these two pairs. Hubs (or concentrators, routers, etc.) are commonly seen as facilities in which some commodity (such as mail, airline passengers, etc.) comes from many different origins. This type of network is

[1] Other examples commonly used are associated to public transportation systems and telecommunications networks.

often called a hub-and-spoke network. These movements (called flows) are grouped and re-routed to their destinations, sometimes via another hub. Thus, all the flow that is exchanged between nodes on the network has to be routed via the hubs to which the nodes are attached.

Depending on how nodes are allocated to the hubs, two types of hub networks can be distinguished: single allocation and multiple allocation. In single allocation networks, the flow originated and destined to each node is routed through only one hub. So, each node can only be allocated to one hub. In multiple allocation networks, the incoming and outgoing flow to each node can be routed through more than one hub.

Hub location problems can be of various types, depending on the objective function as well as on considered constraints. Concerning the objective function, some examples are the p-Hub location problem, in which the number of hubs to be opened is fixed and is known (to a value p) and the objective is to minimize the total flow cost plus the opening cost of the hubs (fixed costs of opening the hubs can be removed from the objective function, resulting in a p-Hub Median Problem) or the p-Hub Center Problem, a minimax type problem where the objective is to allocate the nodes to chosen hubs in such a way that the maximum path between any origin/destination pair is minimized, as defined by Campbell [20]. Concerning the constraints, we can discriminate the capacity restrictions applied to the flow on each arc, capacity limits to the total/incoming/outgoing flow that each hub can handle or, as already mentioned, fixing a priori the number of hubs to be opened to a value p.

More recently, Correia *et al*. [26] proposed a new extension of the HLP: the Multi-Product Capacitated Single-Allocation Hub Location Problem (MP-CSAHLP). In this extension, several kinds of flows are considered, including when multiple types of products are to be shipped between the origin/destination pairs. In this problem, the hubs can be classified as exclusive (hubs handle only one type of flow) or non- exclusive (hubs handle more than one type of product). Being a capacitated problem, there are capacity constraints for the incoming flow to each hub. Also, the hub network is complete for each product although, when considering the hub network as the entire network, it does not necessarily have to be complete.

Figure 2 shows an example of a HL network where we can see an example of a flow and the associated collection, transfer and distribution rates between two nodes passing by the connected hubs. These rates (or discount factors) are common associated with the distances and simulate the economy of scale.

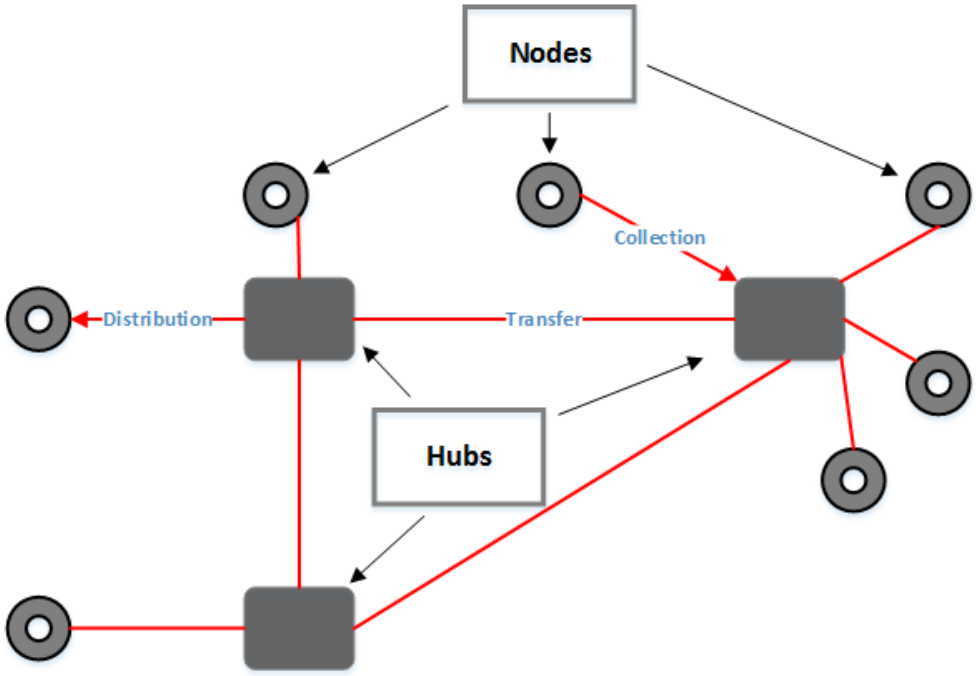

Figure 2: Hub Location problem example.

HLP has received a huge attention and exact and heuristic methods have been proposed to tackle it. More information about Hub Location models is available at [4,19,41,91,95].

Goldman [59] was the first to address the location of hubs, but perhaps Hakimi [64], in the paper "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph", was the first to introduce the location of switches in a network. In the 80's, O'Kelly [92,94] proposed the first mathematical representation for the problem, followed by Campbell [20] who proposed the first linear programming formulation for the single and multiple allocation and for the Capacitated and Uncapacitated versions of the problem.

Bellow we will present the state of the art for the CSAHLP and the CSApHLP.

2.1.2. Capacitated Single Allocation Hub Location Problem

One of the HLP extensions with limited capacity on the hubs is the Capacitated Single Allocation Hub Location Problems (CSAHLP). The term single allocation means that a node only can be connected to one hub and being capacitated means that the hub has a fixed capacity in terms of flow (sum of all the nodes connected to it either as origin or destination) and also has a fixed cost of acting as a hub. The goal of this problem is to decide which nodes are to be used as hubs (the number of open hubs is not known) and determine the allocation of the other nodes to the hubs, in order to route the flows between these two pairs (the route of flow must be made by at least one and at most two hubs) at minimum cost.

Numerous exact and heuristic techniques were proposed for the CSAHLP over the past years with competitive results. A summary of some of the latest papers and surveys is presented in Table 2.

Heuristic/Lagrangean Relaxation thecnique	[117] Parallel branch and bound with local search (2015) [84] Variable Neighborhood Search (2013) [118] An Ant Colony System with a Genetic Algorithm (2012) [23] Lagrangean Relaxation (2009) [89] Genetic Algorithms (2009) [99] Ant colony (2008)
Exact	[117] Parallel branch and bound with local search (2015) [114] Sub-problema approach with CPLEX (2014) [75] Branch and Cut (2005)
Surveys	[41] Hub location problems: A review of models, classification, solution techniques, and applications (2013) [27] The capacitated single-allocation hub location problem revisited: A note on a classical formulation (2010) [4] Network hub location problems: The state of the art (2008) [93] Hub network design with single and multiple allocation: A computational study (1996)

Table 2: Summary of the latest papers and surveys on CSAHLP.

A Branch & Bound [80] technique is found in the work of Ernst and Krishnamoorthy [36] to solve the CSAHLP, in which the authors present a modified formulation of the CSApHLP (where p is the number of necessary hubs to be opened) to solve the CSAHLP. They were the first authors to present computational results for this problem, testing the algorithm for small and large instances. Correia et al. [27] explored the problem formulation leading to the inclusion of additional sets of constraints that helped to decrease the computational time needed to solve the problem to optimality, for the smaller instances. Labbé et al. [75] examined its polyhedral properties and produced a Branch & Cut algorithm for this problem. Costa et al. [28] presented a second objective function to the model, together with an interactive procedure to generate non-dominated solutions. Computational results are based on a comparison between single and double objective functions. Almeida et *al*. [2] proposed a Local Branching to solve the CSAHLP. This method is based on Branch & Cut together with a heuristic and local search techniques to explore solution neighborhoods. Stanojević and Marić [114] developed an algorithm to solve both the CSAHLP and the USAHLP (the uncapacitated version). The main idea is to search the entire hub configuration, and for each one, an exact approach (with the commercial solver CPLEX) is used to solve the allocation sub problem. They have tested the algorithm for small and large instances with very good results, but with expensive computational time.

Besides a Branch & Bound approach, Ernst and Krishnamoorthy [36] presented a random descent and a simulated annealing [73] algorithms for this problem. Population based heuristics can be found in the work of Stanimirović [115], Mohammad [89] and Almeida et *al*. [3] using Genetic Algorithms [7], and Ant Colony optimization [32] on Randall [99]. Chen [21] presents an heuristic procedure to solve the CSAHLP. He assigns 3 levels to the problem. The first level is the determination of the number of hubs required; in the second level chooses the location of the hubs, given the number of hubs found in first level; the third level is the allocation for the hubs found in the second level. With this approach, Chen achieved very good results for small and large instances, within reasonable computational time. Stenojević et *al*. [117] proposed a hybrid procedure, combining a parallel Branch & Bound and an evolutionary algorithm

to solve this problem, with good results. Contreras et *al*. [23] introduced a Lagrangean Relaxation and reduction tests based on bounds that reduced the size of the problem and consequently the computational time. More recently, Marić [84] presented a Variable Neighborhood Search [88] for choosing the number of open hubs and its locations, and CPLEX to solve the allocation part, knowing the open hubs. Computational results for small instances are presented.

2.1.3. Capacitated Single Allocation p-Hub Location Problem

One of the HLP extensions (with limited capacity on the hubs and a predefined number of hubs to be located) is the Capacitated Single Allocation p-Hub Location Problems (CSApHLP). As for the CSAHLP, the term single allocation means that a node can only be connected to one hub, and being capacitated means that the hub has a fixed capacity in terms of flow (sum of all the nodes connected to it either as origin or destination). There is a fixed cost for being a hub and the number of nodes acting as hubs in known. The objective consists of choosing a set of p nodes to be established as hubs and minimize the total cost of the allocation of all non-hubs to the chosen hubs that does not violate the capacity constraint of the hubs.

Despite this problem has not been as studied as the CSAHLP, more recently has raised much attention in the scientific community. There are some articles in the literature that deal with the capacitated version of this problem but don't consider the fixed cost of opening a hub, making the problem simpler to solve (called Median Problems). We address the difficult version, that is, the Capacitated Single Allocation *p*-Hub Location Problem, which deals with the fixed cost of opening a hub in certain locations (a possible formulation of this problem is presented in formulation 20 to 32). Below is presented an up-to-date literature review concerning the Hub Median Problems and then we will address the Hub Location Problems.

O'kelly [94] presented this problem and derived the first quadratic mathematical formulation, proposing also two heuristics and presenting computational results with 2, 3 and 4 open hubs. Klincewicz [70] presented and compared various interchange heuristics computationally. Campbell [18]

presented integer formulations for the multiple (UMApHMP) and the single p-hub median problem (uncapacitated versions) and also presented two heuristics to tackle these problems. Ernst [37,38] also presented an exact and heuristic methods for solving both USApHMP and UMApHMP, that is, only for uncapacitated versions with the number of hubs already known.

Perez *et al*. [97] introduced a hybrid GRASP-Path relinking to solve this problem where GRASP [43] is used to construct the population for the Path Relinking algorithm [52]. They presented results for the AP dataset [8] (standart dataset in literature for this problem) and solved instances up to 100 nodes and p between 2 and 20. Stanimirović [116] proposed a Genetic Algorithm approach for dealing with this problem but didn't consider fixed costs of choosing a node to act as a hub in a given location. The author reached optimal solutions with his approach in AP dataset for instances up to 50 nodes and also solved larger instances up to 200 nodes. We tackle the more difficult version of considering fixed costs related to open a hub in a specific location.

Concerning the problem that we address, the CSApHLP, only two articles can be found regarding this problem, making it a fairly new problem. Lu *et al*. [83] produced a Lagrangean Relaxation procedure with subgradient optimization. Basically, this procedure divides the problem formulation into two sub-problems (as in the work of Contreras *et al*. [23]): the space of Z variables, and the space of X variables. Using the commercial solver Gurobi, the Z space variables are solved, and for solving the X space variables (the assignment problem) they use a simple procedure to find the shortest path between nodes.

The author presents computational results for a known data set dealing with instances up to 50 nodes and for p=2, 3, 4 and 5, and also presents results using Gurobi for some instances.

The latest article found addressing this problem is a reactive GRASP proposed by Ting *et al* [122]. The general GRASP [43,44] procedure as three basic procedures:

1. the construction phase, where the local search (or a greedy method) produces a solution (here, a restricted candidate list is created);

2. the improvement phase, where the solution given in (1) is improved with an improvement method (such as tabu search);
3. solution update method (where the solution obtained with (2) is updated by the best solution found so far, if it is better than that one).

This procedure goes from (1) to (3) till some termination criteria is achieved. In reactive GRASP, proposed by Prais and Ribeiro [96], the first parameter (construction of the restricted candidate list) is self-adjusted according to the solution quality previously obtained.

The author provides computational results for the AP data set and solve instances up to 50 nodes and for p=2, 3, 4 and 5, and also presents results using Gurobi software optimizer for some instances. As far as we know, in the literature, there are only results for instances up to 50 nodes.

A summary of the latest papers and surveys is presented in Table 3.

Heuristic/Lagrangean Relaxation thecnique	[122] Reactive GRASP (2015) [83] Lagrangean Relaxation (2013) [116] Genetic Algorithm
Exact	[83] Lagrangean Relaxation using GUROBI (2013)
Surveys	[41] Hub location problems: A review of models, classification, solution techniques, and applications (2013) [4] Network hub location problems: The state of the art (2008) [47] Hub Location Models in Public Transport Planning (2008) [70] Theory and Methodology: Heuristics for the p-hub location problem (1991)

Table 3: Summary of the latest papers and surveys on CSA$\boldsymbol{p}$HLP.

3. Relaxation Adaptive Memory Programming

The RAMP (Relaxation Adaptive Memory Programing) method is a metaheuristic approach proposed in 2005 by Cesar Rego [100], based on the exploration of the relationship between the primal and dual sides of the problem, for solving complex optimization problems. The search is based on adaptive memory principles on the primal sides and relaxation techniques on the dual side. In the next two subchapters, adaptive memory programming concepts and relaxation techniques will be introduced, being these the main components that comprise the RAMP method for the exploration of the dual and primal sides. The last subchapter is for describing the general RAMP framework.

3.1. Adaptive Memory Programming

Adaptive memory programming (AMP) [53,79,121] has been the basis of numerous important developments in metaheuristics in the last two decades. The AMP refers to the integration of memory structures in optimization algorithms so it can explore more intensively and effectively the solution space of such problems. Tabu Search (TS) [54] was perhaps the first technique to use AMP in the search procedure but more recently other methods use adaptive memory mechanisms [121] such as Scatter Search [50], Genetics Algorithm [111] and others. For the sake of simplicity this principle will be explained with TS framework since that this principle was born with it in the late 80's.

Fred Glover introduced in 1986 a metaheuristic called Tabu-Search (TS) [51]. TS allows non-improving methods, i.e., the deterioration of the current solution, but preventing cycling back to previously visited solutions by the use of memory strategies [48,51]. Tabu lists (short-memory) prevent the algorithm from visiting the same solutions in the last *n* moves. Sometimes this approach can be inconvenient since moving to a previous visited solution can improve our solution if afterwards we follow a different path, but this is considered a forbidden move. To prevent this type of situations, usually there is an aspiration criterion that allows a temporary "ignore" of the tabu list (the simpler

aspiration criterion is when a prohibited move allows us to improve the best solution found so far).

Figure 2 describes a general procedure for the tabu search basic implementation. Generally, for each iteration the algorithm computes a new solution based on some move. If that move is not in the tabu list (or if the best solution found is improved) the algorithm continues, otherwise it chooses the best move that is not in the tabu list. The algorithm terminates when some termination criteria is achieved, generally one of the following: “after a fixed number of iterations or after a fixed number of iterations without improvement, or when the objective function reaches a pre- specified threshold value” [48].

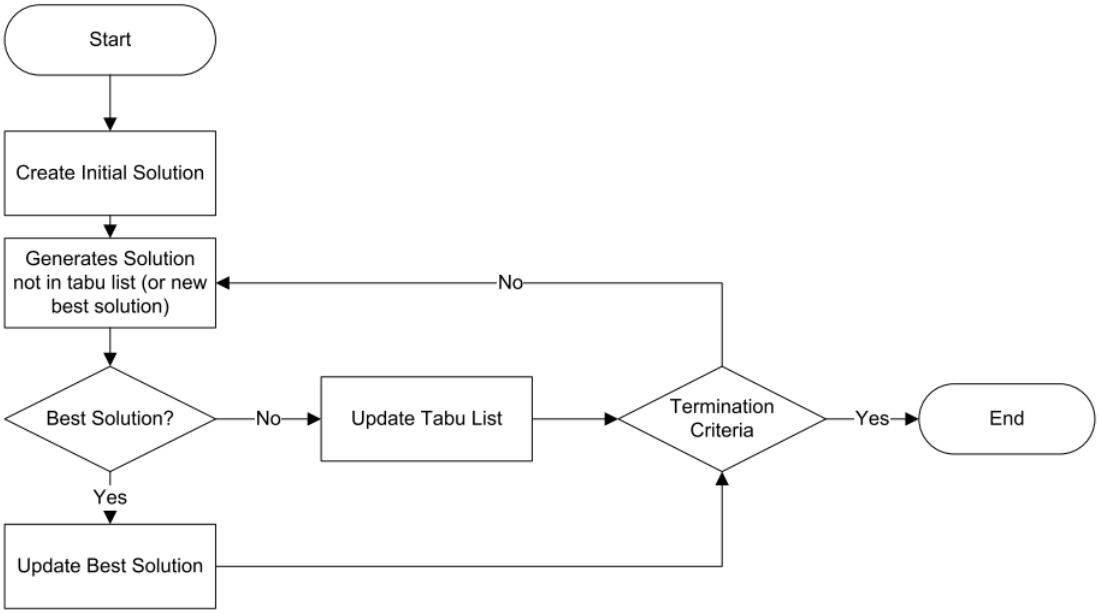

Figure 3: Tabu Search Generic Procedure

The algorithm has other principles that also use memory structures: Intensification and Diversification. Intensification allows the algorithm to search spaces where the possibility of achieving better solutions is higher, and Diversification allows searching spaces less visited, which could improve the objective function value. AMP is implicitly linked with this metaheuristic, because the premise of TS is preventing the algorithm from revisiting solutions obtained in previous iterations with short term memory. The AMP in TS acts as a learning process and can be characterized by frequency, statistically or other

simple or complex approach regarding the solution quality in the searching process.

The first publication of this technique involved only short-term memory: the tabu list. Later, the author published the remaining TS procedure [51,55] with the use of long term memory structures. The exploration of the solution space is guided based on the choice of a search direction on the attributes of the current solution and on historical data. The Tabu Search allows different levels of sophistication depending on the type of memory used and the strategy chosen for the guiding process. If an intensification procedure is used, the search space is explored more intensely along the feasible solutions region that have been identified as promising; for example, using long-term memory to identify common features in the best solution found, and if so, initiate an intensified search phase in which these solutions with these characteristics (or restricting the neighborhood to such solutions) are preferred. Diversification process is other component of TS and it is used to force the search to explore parts of the feasible solution space that have not yet been explored; for example, by penalizing solutions with common features to recently visited solutions.

Scatter Search (SS) [77,86] and Genetic Algorithms (GA) [111] are other frameworks that embody AMP principles. For these techniques, the memory is constituted by populations of solutions and can incorporate tabu search components. For example, SS uses and classifies the use of memory as “inheritance memory”. Basically, the fourth method of SS (Subset Generation Method) is for keeping track of subsets of solutions that have been subjected to the combination (the fifth method) mechanism from one iteration to the next (the first method is diversification and then the improvement followed by the reference set update method). When new solutions are admitted to the reference set, the method generates only those subsets that are admissible for combination in the current iteration. The Subset Generation Method performs this operation by a memory structure that identifies the subsets containing new reference solutions. In one way or the other, memory principles are used in a variety of procedures that demonstrates its potential to solve effectively several optimization problems. As it was said in the beginning of this chapter,

RAMP encompass memory principles as expressed in Tabu Search for the exploration of primal space together with dual solution space exploration.

3.2. Relaxation Techniques

Different relaxation techniques have been widely used in the field of combinatorial optimization giving lower and upper bounds for the problems to be addressed. Many heuristics have been implemented based on relaxation techniques, producing competitive computational results. The main advantage of using mathematical techniques relaxation is related to the fact that the resulting problem is usually easier to solve and its resolution provides limits (upper or lower) for the value of the objective function of the original problem.

During this thesis, various types of mathematical relaxations are used. Below, a summary of techniques and key concepts based on the description provided in [100] are presented.

In the description that follows, it will be used the following definition of a generic integer linear programming problem $0-1$, P:

$$\begin{aligned} &(P) = Min\ cx \\ &\text{s.a.} \\ &\quad Ax \leq b \\ &\quad Dx \leq e \\ &\quad x \in \{0,1\} \end{aligned} \tag{35}$$

where it is assumed that constraints that make the problem difficult to solve are represented by $Ax \leq b$. That is, if we exclude these constraints in the original problem, it can be solved effectively by known methods.

The ***Lagrangean Relaxation*** (LR) [10,49,62] is one of the most widely relaxation techniques used to solve complex optimizations problems and was proposed by Everett [39] in the early 70's. The Lagrangean Relaxation P is obtained relaxing the constraints $Ax \leq b$. The new problem LP^{λ}, is defined by:

$$
\begin{aligned}
& LP^{\lambda} = Min\ cx\ +\ \lambda(Ax\ -b) \\
& \text{s.a.} \\
& \quad Dx \leq e \\
& \quad x \in \{0,1\}
\end{aligned}
\tag{36}
$$

where λ is a non-negative vector of multipliers with an element for each line of A. In this case, the solution of the dual problem is to find λ that maximizes $v(LP^{\lambda})$, that is, the value of the objective function LP^{λ}.

An LP^{λ} solution for a given vector $\lambda \geq 0$ gives a lower bound for $v(P)$, which represents the value of the objective function of the primal problem P (weak Lagrangean duality). When the optimum values for the primal and dual problems differ (you define this difference by duality gap) is not possible to obtain the optimal solution of the integer linear program by solving the dual problem.

The strong Lagrangean duality (which defines optimality conditions for a primal-dual Lagrangean solution[2]) includes conditions for additional deviations (complementary slackness conditions), that is, for a given λ, a primal solution x must satisfy the expression $\lambda(Ax\ -b) = 0$. If a primal-dual optimal solution does not satisfy these conditions, the solution is called a near-optimal solution of P with duality gap $v(P) - v\left(LP^{\lambda}\right) = \lambda(Ax\ -b)$. Determining a primal-dual optimal solution involves finding the optimal multipliers for LP^{λ} resulting from the resolution of the Lagrangean dual of primal problem P which can be defined by:

$$
\begin{aligned}
& D^{\lambda} = Max\ LP^{\lambda} \\
& \text{s.a.} \\
& \quad \lambda \geq 0
\end{aligned}
\tag{37}
$$

[2] A primal-dual Lagrangean solution is an optimal solution to the LP^{λ} problem that also satisfies the $Ax \leq b$ constraints of the primal problem.

Which is equivalent to the problem:

$$\begin{aligned} & D^{\lambda} = Max_{\lambda \geq 0}\, Min_{Dx \leq e}\, cx + \lambda(Ax - b) \\ & \text{s.a.} \\ & \quad x \in \{0,1\} \end{aligned} \tag{38}$$

So, to determine $v(LP^{\lambda})$ for a given λ, it is necessary to solve the Lagrangean problem LP^{λ}.

While Lagrangean approaches incorporate the most tricky constraints into the objective function by creating linear combinations of them, the ***Surrogate Constraints Relaxation*** (SCR) initially proposed by Glover [56], generate new constraints to replace those considered difficult. Basically, given the original problem P, this relaxation consists in replacing the constraints $Ax \leq b$ by a non-negative linear combination of these constraints by using a vector of weights w. Thus, the constraints $Ax \leq b$ are replaced by a single constraint $w(Ax - b) \leq 0$ generating the surrogate problem, SP^{w} defined by:

$$\begin{aligned} & SP^{w} = Min\, cx \\ & \qquad\qquad \text{s.a.} \\ & \quad w(Ax - b) \leq 0 \\ & \quad Dx \leq e \\ & \quad x \in \{0,1\} \end{aligned} \tag{39}$$

The surrogate dual problem can be replaced by:

$$\begin{aligned} & D^{w} = Max\, SP^{w} \\ & \text{s.a.} \\ & \quad w \geq 0 \end{aligned} \tag{40}$$

These procedure is also widely used in mathematical programming applications and produce stronger relaxations for combinatorial optimization than Lagrangean methods [57].

The **Subgradient Optimization** procedure (proposed in [65]) has been widely used for solving the Lagrangean Dual problem (LP^{λ}), and more recently has demonstrated to be very efficient in determining good weights for surrogate constraints. Let λ^* be the optimal solution for the Lagrangean dual D^{λ}. This procedure starts with a certain point λ^0 (for example $\lambda^0 = 0$) and iteratively generates a sequence of points:

$$\lambda^{k+1} = \lambda^k + \Theta^k(Ax^k - b) \tag{41}$$

Where Θ^k is a positive scalar called step size, typically determined in the following way:

$$\Theta^k = \frac{\beta^k\left[v(D^{\lambda}) - v(LP^{\lambda}))\right]}{||Ax^k - b||^2} \tag{42}$$

Where β^k is a parameter between 0 and 2 and $v(D^{\lambda})$ is an upper limit of the optimum value D^{λ}, being x^k an optimum solution to the LP^{λ}. By the weak duality theorem, $v\left(D^{\lambda}\right)$ is replaced by a feasible solution for P and the parameter β^k is initialized with the value of 2, being reduced in half when $v(D^{\lambda})$ has not improved for a fixed number of iterations. In the same way, the method stops when there isn't a substantial improvement over a predefined number of iterations T, for example:

$$\begin{aligned} &\left|v\left(LP^{\lambda^k}\right) - v\left(LP^{\lambda^{k+1}}\right)\right| < \varepsilon \\ &for\ a\ given\ \varepsilon > 0\ and\ 1 \le t \le T \end{aligned} \tag{43}$$

The ***Cross Parametric Relaxation*** [100] combines Lagrangean Relaxation with surrogate constraints using subgradient optimization to generate good surrogate constraints. The subgradient method is used to generate the vector of surrogate multipliers for the relaxed constraints of the primal, to create the corresponding surrogate problem. Then, the surrogate vector is used as a parameter vector in the subgradient search carried out on the Lagrangean Relaxation of the surrogate problem, aimed at determining a surrogate dual

solution. It should be noticed here that if only the surrogate constraint is relaxed the Lagrangean multiplier is a scale factor rather than a vector. However, there are cases where relaxing more than one component, under this Lagrangean substitution framework, may be advantageous. In any event, the new surrogate dual solution and the primal solution (obtained by projecting the surrogate dual solution on the primal- feasible region) yield the new lower and upper bounds, respectively. These bounds, in turn, are used to compute the next subgradient-based multipliers and generate a new surrogate problem, thus completing the loop for one iteration of a parametric subgradient search.

Formally, the cross-parametric relaxation can be defined as follows:

$$\begin{aligned} & L_\lambda SP^w = Min\ cx + \lambda w(Ax\ - b) \\ & \text{s.a.} \\ & Dx \leq e \\ & x \in \{0,1\} \end{aligned} \tag{44}$$

where w is a multiplier vector for the surrogate problem and λ is a scalar representing the Lagrangean multiplier associated with the surrogate constraints.

The corresponding dual problem can be defined by:

$$\begin{aligned} & D^{\lambda w} = Min\ L_\lambda SP^w \\ & \text{s.a.} \\ & \lambda, w \geq 0 \end{aligned} \tag{45}$$

Defining $\varphi = \lambda w$, $D^{\lambda w}$ becomes the Lagrangean dual D^{φ}, therefore identifies the same dual optimal solution.

The motivation for relaxing the problems in hand is also reinforced by the fact that, in several practical situations, problems are so complex that the corresponding formulations may involve a relatively large set of difficult constraints of the type identified as $Ax \leq b$ in our general formulation. Combining the dual space search, using some proper relaxation procedure, with the primal space search, by the use of some heuristic such as Tabu Search,

Scatter Search, or other, together with some adaptive memory structures, are the foundations of RAMP framework that is a very competitive metaheuristic to solve efficiently and effective many combinatorial optimization problems.

3.3. RAMP Framework

The RAMP (Relaxation Adaptive Memory Programing) method is a metaheuristic approach presented in 2005 by Cesar Rego [100], based on exploring the primal-dual relationship, for solving complex optimization problems. The search is based on adaptive memory principles in the primal side and relaxation techniques on the dual side. The term primal relates to the primal solution space of the original problem while the term dual refers to the dual problem of a relaxation of the original problem. The method allows different levels of sophistication, depending on the intensification effort in the exploration of primal and dual sides, and techniques used to combine the best solutions obtained through both components (primal and dual). In its simplest form, named Dual-RAMP, this method operates more prominently in the dual side (or phase). Higher level of sophistication, named PD-RAMP, allows more exploration of the primal side, incorporating more complex memory structures.

We can define in general terms the RAMP method as combining concepts of Adaptive Memory Programming and search techniques with relaxation mathematical principles, scanning the primal and dual solutions space in order to obtain crucial information about the problem. The use of Lagrangean Relaxation and surrogate constraints, among other relaxation techniques (or cross parametric relaxation proposed in RAMP paper), are examples of dual side strategies. As primal sides strategies, Scatter Search or Path Relinking [50] are examples of appropriate methods to use with the concept of adaptive memory, but other procedures can be used in primal exploration. In general terms, the RAMP method can be described as an incremental process, starting with the simplest level of sophistication and passing to successively more complex forms if the results are not the expected ones.

Figure 4 shows the generic model of the PD-RAMP framework. It represents a general design overview of the framework that must be adapted to the sophistication level in use and to the problem being considered. It exemplifies

the most sophisticated level of the framework: PD-RAMP. In this example, the relaxation method (Dual Phase) is the Cross Parametric Relaxation Technique. The Restrictive Method (Primal Phase) is a Scatter Search, but naturally other methods (evolutionary methods preferentially) can be applied both on the Dual or Primal phase of the problem. When the method only concentrates on the Dual phase of the problem we have the framework's simplest sophistication level: the Dual-RAMP.

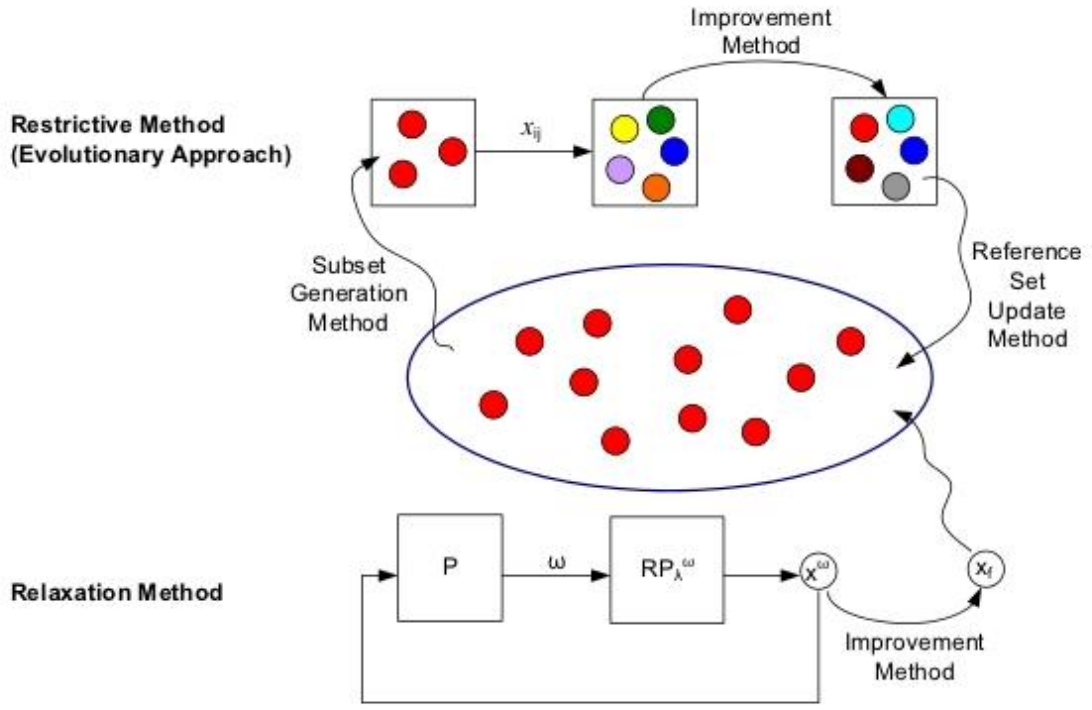

Figure 4: RAMP general approach (image taken from [100])

Thus, the PD-RAMP approach explores intensively both sides of the problem: the Dual and the Primal. The Dual-RAMP is integrated into the PD-RAMP and the solutions obtained by the Dual-RAMP approach are incorporated into the reference set, providing valuable information for the problem resolution. In the above example (Figure 4), Scatter Search extracts a small subset from the reference set, and generates new solutions from the subset.

An improvement method is then applied to these solutions and integrated into the reference set if they satisfy a given condition[3].

3.4. Previous Applications

There are already some Dual-RAMP and PD-RAMP applications that originated high quality results for hard combinatorial optimization problems, such as the Resource Constraint Project Scheduling [106], the Uncapacitated Facility Location Problem [45], the Multi-Resource Generalized Assignment Problems [110] and the Capacitated Minimum Spanning Tree Problem [101].

The RAMP approach for the Uncapacitated Facility Location Problem (UFLP) involved a Dual-RAMP approach. For this problem, the most basic level of sophistication, the Dual-RAMP, was sufficient to obtain excellent results, when compared with the best results known from the literature. For this implementation was used, on the dual side, the Dual Ascent Method from Erlenkotter's DUALOC algorithm [35] and on the primal side the improvement method was based on the Tabu Search approach proposed by Michel and Hentenryck [87]. The results obtained by RAMP algorithm on several benchmark instances, were compared with the best-known algorithms for the UFLP: two Tabu Search approaches from Michel and Hentenryck [87] and Sun [119], and another approach from Resende and Werneck [103], a hybrid algorithm based on a greedy algorithm and path relinking.

The results obtained by the RAMP approach outperformed all other approaches producing optimal or near optimal solutions for all instances in reasonable computational times. This is a general overview of this framework, and the final RAMP model for a specific problem depends on the strategy defined. As already stated, there are numerous RAMP applications, all producing state of the art algorithms for several different problems. Despite RAMP is not a widely-known metaheuristic, the quality attained with previous RAMP applications, anticipates a great potential for RAMP applications to other complex combinatorial optimization problems.

[3] As example, we can define the reference set composition with only the best solutions found, and scatter search will replace solutions from the reference set, only if a new best solution is generated.

4. Overview of Papers

This chapter introduces a brief introduction to each paper of this thesis, which are provided in full length in *Scientific Paper* chapter.

4.1. RAMP Algorithm for the CFLP

This paper presents a RAMP algorithm for solving the well-known Capacitated Facility Location Problem. The RAMP algorithm for CFLP was based on the simplest version of the RAMP method, in which only the dual side is explored more intensely, producing competitive results with the best algorithms. In the dual side was used an Lagrangean Relaxation technique with subgradient optimization procedure, and a tabu search was used to explore the primal side of the problem. The proposed algorithm was experimented on standard benchmark (small, median and large dataset) and compared with the best-known algorithms in the literature and it was verified that the proposed algorithm obtained state of the art results.

This work was presented in:

- 27th European Conference on Operational Research, 2015.
- MIC 2015: The XI Metaheuristics International Conference, 2015.

This work was submitted for presentation on:

- 8th International Conference on Computational Logistics – ICCL'17 that be held on 18–20 October 2017 in Southampton, United Kingdom

4.2. RAMP Algorithm for the CSAHLP

This paper introduces a RAMP algorithm for solving the Capacitated Single Allocation Hub Location Problem. The RAMP algorithm used was a Dual-RAMP witch strongly prevails the dual side exploration. The RAMP algorithm combines a Lagrangean Relaxation technique with a local search procedure to explore primal-dual relationships as a way to create advanced memory structures that integrate information from both primal and dual solution spaces. The algorithm

was tested in the well-known AP dataset with excellent results. Comparison against other state of the art algorithms were made proving that the RAMP framework is an exceptional metaheuristic to tackle these very difficult classes of problems.

This work was submitted and accepted for presentation and will be included in the conference proceedings of:

- The 17th International Conference on Computational Science and Its Applications (ICCSA 2017) that be held on July 3 - 6, 2017 in Trieste, Italy.

4.3. RAMP Algorithm for the CSApHLP

This paper presents a RAMP algorithm for solving the Capacitated Single Allocation p-Hub Location Problem, which is a variant of the CSAHLP but with the number of hubs known. With the results achieved for the CSAHLP, it was expected that a RAMP approach to this problem would produce also very good results. A Dual-RAMP algorithm was used to strongly explore the dual side of the problem, combining Lagrangean Relaxation technique with Subgradient search with an improvement method. The algorithm was tested on the well-known AP dataset with excellent results for different number of open hubs. Although the results obtained with the simple version of the framework have been extremely competitive, it was considered to implement a more sophisticated RAMP algorithm for the resolution of the CSApHLP, to evaluate if it could produce even better results. A PD-RAMP algorithm was made where it explores, in the dual side, the same relaxation technique of the Dual-RAMP. The PD-RAMP algorithm explores more intensively the primal side of the problem, with the use of a Scatter Search, allowing a wider searching area. Comparison against state of the art algorithms were made, outperforming all these algorithms with new best solutions found.

Once again, the RAMP framework manages to solve effectively and efficiently this problem, proving that it can produce outstanding results not only for FLP but also for HLP, where it rivals with the best algorithms in the literature.

This work was submitted and accepted for presentation and will be included in the conference proceedings of:

- 47th International Conference on Computers & Industrial Engineering (CIE47) hosted and organized by Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa, Portugal, on October 11th -13th, 2017.

5. Conclusions

Facility Location and Hub Location problems play an important role in Operational Research and have been widely discussed in the literature, having numerous applications in the real world. Several exact and heuristic algorithms have been proposed to solve effectively and efficiently these problems, producing state of the art results.

Metaheuristics play an increasingly important role in obtaining good solutions in reasonable computational time, unlike exact algorithms, which require great computational effort to produce optimal solutions. Being extremely difficult problems (NP-hard), metaheuristics have been predominant in addressing them, with increasingly different approaches and methods for obtaining near optimal results.

Different algorithms were proposed in this thesis with a common denominator: the RAMP framework. For CFLP, the combination of Lagrangean Relaxation and Tabu Search with the use of memory structures, led to excellent results in low computational time, rivaling with the best algorithms in literature. Regarding the two versions of HLP, a simple and both simple and advanced level of sophistication of the RAMP procedure applied to CSAHLP and CASpHLP, respectively, has produced state of the art results, with new best-known solutions obtained for standard datasets, therefore outperforming the best algorithms known until now. The combination of mathematical relaxation techniques (and in particular the Lagrangean Relaxation) with adaptive memory techniques, and its interconnection with appropriate search algorithms, have proven to be a highly effective and efficient approach for solving such difficult problems. In a general way, all the algorithms proposed in this work are capable of generating good solutions in a reasonable computational time.

In the next subchapters, it will be introduced a short list of main contributions and possible directions for future research in FLP and HLP.

5.1. Contributions

The research question proposal was built on recently developed original work that aimed to investigate the application of the RAMP approach to a variety of fundamental combinatorial optimization problems, for which current state of the art optimization algorithms compete to find high quality solutions.

The recent success of RAMP applications to a wide variety of hard combinatorial problems suggested that an application to capacitated FLP and HLP variants would produce competitive results, challenging the best-known algorithms for the solution of those problems. This would be the main contribution of the present work. The success obtained with different RAMP application for each of the presented papers has indeed validated that this is a very effective framework, as it has:

- Successfully applied the RAMP algorithm to the CFLP, producing excellent results in very reasonable computational time and rival with the best algorithm in the literature;
- Effectively applied the RAMP algorithm to the CSAHLP, contributing with new best-known solutions for standard datasets;
- Efficiently applied two RAMP algorithms (with different levels of sophistication) to the CSApHLP, a problem that is relatively new in the literature, and where RAMP played an important role in producing results for standard datasets and given new best-known solution to the scientific community.

5.2. Future Research

The accomplishment of the results obtained with the proposed algorithm suggests that this approach can be applied to other difficult problems (and relevant applications) of combinatorial optimization, with the same degree of success. In particular, the main directions for future research with the RAMP approach lies on the development of new algorithms for tackling several FLP and HLP problems.

Concerning the FLP, the SSCFLP (Single Source Capacitated Facility Location problem) has an identical formulation to the CFLP, and the fact that

the proposed approach is very competitive with existing literature suggests similar results for the SSCFLP. The significant differences of this problem for the CFLP are that each customer can only be served by a single facility, while in the CFLP it may be served by several facilities.

The Uncapacitated HLP with single assignment is also a problem that is requiring a lot of attention from the scientific community. The results obtained for both versions, leads one to think that an application of RAMP to the USAHLP (Uncapacitated Single Allocation Hub Location Problem) or even to the USApHLP (same as USAHLP but with a predefined number of open hubs) can produce state of the art results. The only difference to the tackled problems is that in this case, the hubs do not have a capacity.

Other extensions of HLP, in which it is expected that RAMP produces good results, are the CMAHLP or the CMApHLP (same as CMAHLP but with a predefined number of open hubs) or even the UMAHLP or the UMApHLP (same as UMAHLP but with a predefined number of open hubs). In these cases, the approach should be extremely modified in order to consider the multiple allocation that a node can have. Nevertheless, it is speculated that given the success obtained with both versions of HLP, a RAMP application to one of these problems may be obtain results with the same level of quality than those obtained for the problems addressed in this thesis.

Scientific Papers

6. Dual-RAMP for the Capacitated Facility Location Problem

Abstract: Facility location embodies a class of problems concerned with locating a set of facilities to serve a geographically distributed population of customers at minimum cost. We address the classical Capacitated Facility Location Problem (CFLP) in which the assignment of facilities to customers must ensure sufficient facility capacity and that each customer is served by only one facility. This is a well-known NP-Hard problem in Combinatorial Optimization that has been extensively studied in the literature. Due to the difficulty of the problem significant research efforts have been devoted to developing advanced heuristics methods aimed at finding high-quality solutions in reasonable computation times. We propose a Relaxation Adaptive Memory Programming (RAMP) approach for the CFLP. Our method combines Lagrangean subgradient Search with Tabu Search to explore primal-dual relationships and create advanced memory structures that integrate information from both primal and dual solution spaces. The algorithm was tested on the standard ORLIB dataset and on other very large-scale datasets for the CFLP. Our approach efficiently found the optimal solution for all instances in ORLIB, produced very competitive results for the very large instances and found ten new best-know solutions. Comparisons with the current best performing algorithms for the CFLP show that our Dual-RAMP algorithm exhibits excellent results.

Keywords: RAMP, Facility Location, Adaptive Memory Programming, Lagrangean Relaxation.

6.1. Introduction

The Capacitated Facility Location Problem (CFLP) is a well-known combinatorial optimization problem that belongs to the class of the NP-Hard problems [46]. The CFLP can be described as:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} D_j C_{ij} x_{ij} + \sum_{i=1}^{m} F_i y_i \tag{6.1}$$

s.t.

$$\sum_{i=1}^{m} x_{ij} = 1, \;\; j = 1, \dots, n \tag{6.2}$$

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i y_i, \;\; i = 1, \dots, m \tag{6.3}$$

$$x_{ij} \geq 0 \,, \;\; j = 1, \dots, n \;\; i = 1, \dots, m \tag{6.4}$$

$$y_i \in \{0,1\}, \;\; i = 1, \dots, m \tag{6.5}$$

where m represents the number of possible locations to open a facility and n the number of customers to be served. S_i indicates the capacity of facility i and F_i the fixed cost for opening that facility. D_j represents the demand of client j and C_{ij} the unit shipment cost between facility i and customer j. The variable x_{ij} denotes the amount shipped from facility i to costumer j and y_i indicates whether facility i is open or not. The objective is to locate a set of facilities in such a way that the sum of the costs for opening those facilities and the transportation costs for serving all customers is minimized.

Given a set of open facilities ($y_i = 1, i \in I$), the CFLP has the particularity of becoming a Transportation Problem (TP) that can be solved in polynomial time. The TP can be formulated as:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} D_j C_{ij} x_{ij} \tag{6.6}$$

s.t.

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i, \;\; i = 1, \dots, m \tag{6.7}$$

(2) and (4)

Where C_{ij} is the unit shipment cost from facility i to costumer j, x_{ij} is the amount sent from facility i to costumer j, S_i is the capacity of facility i and D_j is the demand of costumer j. The objective is to determine an optimal transportation scheme between the facilities and customers so that the transportation costs are minimized.

If we eliminate variable S_i from equations (3) and if we set $D_j = 1$ in equations (6.2), we obtain a formulation of the Uncapacitated variant of the problem (UFLP), also widely studied in the literature.

6.2. Related Work

The CFLP problem has been extensively studied over the past 50 years resulting in a large number of algorithmic approaches based on exact and heuristic methods. In 1983 Jacobsen [67] presented an heuristic for the CFLP that extended Kuehn and Hamburger's [74] heuristic, originally proposed for the uncapacitated variant of the problem (UFLP). This heuristic is composed by two phases: an ADD method, which starts with all facilities closed and then iteratively opens the facility that achieves the maximum total cost reduction. This phase ends when no more facilities cause the reduction of the total cost if opened. The second phase consists of a local search method in which an open and a closed facility change their status if this operation reduces the total cost. Relaxation techniques have been frequently used to solve the CFLP. Cornuejols *et al*. [25] proposed several algorithms based on mathematic relaxation to solve large CFLP and presented a comparison between several types of relaxations. Guignard and Spielberg [63] presented a Dual Ascent procedure for the CFLP based on the approaches initially proposed by Bilde and Krarup [13] and Erlenkotter [35] for solving the UFLP. Avella *et al*. [6] proposed a Lagrangean Relaxation for the CFLP, that selects a subset of "promising" variables to form the core problem, and uses an exact (Branch-and-Cut) algorithm to solve this core problem. Lorena and Senne [82] and Beasley [10] obtained good upper and lower bounds based on a relaxation technique that provided good results for the CFLP. Sridharan [113] proposed the use of a cross decomposition method, initially proposed by Van Roy [108], in which the main idea is the exploration of the primal and dual sub-problems with a single decomposition procedure in

order to produce tight bounds, lower and upper. Bornstein [14] proposed an ADD/DROP algorithm (that extends the ADD procedure mentioned earlier) and uses the DROP method to close facilities if this improves the objective function. Bornstein and Azlan [15] proposed the use of reduced tests and dominance criteria to solve the CFLP. They define priorities in changing the status of facilities (open-close or close-open) and use simulated annealing, when the status of the facilities cannot be determined, as a local search heuristic technique.

Exact algorithms such as Branch & Price proposed by Klose and Görtz [72] were also introduced for the solution of the CFLP. Branch & Price is based on Branch & Bound and column generation, and can be explained as a "divide and conquer" method, that divides the problem in a master problem (solved by column generation) and a set of subproblems easier to solve. Ming-Che Lai *et al*. [78] used Benders Decomposition [12] and Genetic Algorithms (GA) for the main problem. The main idea is to divide the original problem in easier subproblems. The original problem is relaxed and the constraints and variables that compose the relaxed problem are divided and solved in separate, alternating between the divided problem and the original problem. The GA procedure is used in the original problem. Elite solutions (populations) are generated, followed by the application of local search and selection and mutation factors, to generate new populations of solutions.

Metaheuristics like Tabu Search (TS) [51,58] and GRASP (greedy randomized adaptive search procedures) [43] have been applied to many combinatorial optimization problems usually obtaining high quality results. Minghe Sun [120], proposed a TS algorithm for the CFLP that provided state-of-the-art results. Sun uses flexible memory structures to search in regions of the solutions space that can be promising. These regions are kept in long-term memory to be intensively explored. Sun uses diversification and intensification strategies and a local search method based on changing the status of facilities (open-close or close-open). When a set of open facilities is found (feasibility is met), a Network Flow method based on Kennington [69] is used to solve the transportation problem. Silva [107] proposed a hybrid GRASP with different parameters (simple and reactive GRASP) that produced good results in reasonable computational time.

A new algorithm was recently proposed by Guastaroba and Speranza [60] to solve the CFLP. This algorithm (kernel search) is a heuristic framework based on the idea of identifying subsets of variables and solving a sequence of MILP (mixed integer linear programming) problems, each of them, restrained to one of the identified subsets of associated variables. For more information about the kernel search and its applications, please refer to [60] and [61].
Firefly (with Genetic Algorithms) by Rahmani and Mirhassani [98] (the Firefly approach is motivated by the social behavior of fireflies and the phenomenon of bioluminescent communication), Ant Colony by Venables and Moscardini [123] and Bee Colony by Levanova and Tkachuk [81] are other recent metaheuristics based on nature observation that produced good results for the CFLP.

6.3. RAMP Algorithm for the CFLP

The Relaxation Adaptive Memory Programming (RAMP) is a relatively recent metaheuristic framework proposed by Rego [100]. The method has proved to be extremely effective in finding optimal and near-optimal solutions for a variety of combinatorial optimization problems, such as, the capacitated minimum spanning tree [101], the linear ordering problem [45] and the resource constrained project scheduling problem [106], among others. The underlying feature of the RAMP method is the creation of advanced memory structures derived from primal-dual relationships and their use in guiding the algorithm search process. This is achieved by exploring the dual solution space of an associated relaxation problem and its primal counterpart through adaptive memory local search and evolutionary methods. The RAMP method may be implemented using different levels of sophistication. In the simplest version of the RAMP framework (Dual-RAMP) we will have an algorithmic approach that combines dual search with a local search based adaptive memory method (such as tabu search or path-relinking) to explore the primal search space. Primal and dual procedures are interconnected by memory structures used to guide the search at a higher level. At a higher level of sophistication, the RAMP method may be extended with an evolutionary approach to implement a Primal-Dual

RAMP (PD-RAMP) approach for exploring primal-dual relationships more extensively.

We propose a Dual-RAMP algorithm for the CFLP that uses a Lagrangean Relaxation [30] on the dual side and a simple tabu search improvement method on the primal side. The local search is based on the classical ADD/DROP neighborhood structure [14].

6.3.1. Dual Method

The Dual Method of our algorithm uses a Lagrangean Relaxation based on Daskin *et al.* [30] that uses subgradient optimization to explore the dual side of the problem. At each iteration, we obtain a solution of the relaxed problem, we project that solution to the primal solution space using a Projection Method and we try to improve it with an Improvement Method (Primal Method).

Specifically, if we relax constraints (6.2), we obtain the following optimization problem:

$$min \sum_{i=1}^{m}\sum_{j=1}^{n} D_j C_{ij} x_{ij} + \sum_{i=1}^{m} F_i y_i + \sum_{j=1}^{n} \lambda_j (1 - \sum_{i=1}^{m} x_{ij}) \tag{6.8}$$

$$Z(\lambda) = \sum_{i=1}^{m} F_i y_i + \sum_{i=1}^{m}\sum_{j=1}^{n} (D_j C_{ij} - \lambda_j) x_{ij} + \sum_{j=1}^{n} \lambda_j \tag{6.9}$$

$s.t.$

$$\sum_{j=1}^{n} x_{ij} \leq y_i, i = 1, \dots, m \tag{6.10}$$

$$\sum_{i=1}^{m} S_i y_i \geq \sum_{j=1}^{n} D_j \tag{6.11}$$

(3), (4) and (5)

Constraints (6.10) and (6.11) have been added to strengthen the linear relaxation and to ensure that the total amount of selected capacity is sufficient to respectively serve all of the demands. The objective is to minimize the Lagrangean function (6.9) over the primal decision variables, y_i and x_{ij}, and to

maximize the function over the Lagrange multipliers λ_j. Maximizing the function over the Lagrangean multipliers can be done using subgradient optimization. Lagrangean multipliers are initialized with $\lambda_j = min_i C_{ij}\ \forall j\ \in J$. The solution of the relaxed problem is achieved after solving to optimality the m continuous knapsack problems, one for each facility as we show below (equations 6.12 to 6.14). To solve the m continuous knapsack problems resulting from the Lagrangean Relaxation of the CFLP we used the greedy algorithm proposed by Daskin *et al*. [30] (we will have one such problem for every candidate location, and V_i will be the value of locating at candidate facility i. This quantity will always be non-positive).

$$min\ V_i = \sum_{j=1}^{n} (D_j C_{ij} - \lambda_j) x_{ij} \tag{6.12}$$

$$s.t.$$

$$\sum_{j=1}^{n} D_j x_{ij} \leq S_i,\ \ i = 1, \dots, m \tag{6.13}$$

$$0 \leq x_{ij} \leq 1 \quad \forall\, j \in n\,, \forall\, i \in m \tag{6.14}$$

We would like to select facilities with large negative coefficients in the objective function (6.12) and with small demands so that they consume relatively little of the capacity in constraints (6.13). Therefore, it is considered the following ratio for each demand node:

$$r_{ij}\ = \frac{(D_j C_{ij} - \lambda_j)}{D_j} \tag{6.15}$$

The quantity r_{ij} is the contribution of demand node j to the objective function divided by the demand node j (or the amount of capacity consumed by demand node j). Thus, r_{ij} is the rate at witch demand node j contributes to the objective function per unit consumption of the capacity. Since the assignment variables x_{ij} can be fractional, we can solve the problem given by equations (6.12) to (6.14) using the following algorithm:

Greedy algorithm proposed by Daskin *et al.* [30]

1. Compute values r_{ij} for all demand nodes j.
2. Sort the r_{ij} values in increasing order. Let $[j]$ be the index of the demand node with the i-*th* smallest r_{ij} value.
3. Set $S'_i = S_i * S'_i$ will denote the remaining capacity at candidate facility i. Set $m = 1$. m will be an index of the demand nodes.
4.
   a. Set $x_{[m]i} = \min\{1, \frac{S'_i}{D_m}\}$. Note that $x_{[m]i}$ is the assignment variable for the demand node with the m-*th* smallest value of r_{ij}.
   b. Reduce the remaining capacity site i, S'_i, by $D_m x_{[m]i}$.
   c. Increment m by 1.
5.
   a. If m is less than or equal to the number of demand nodes and $S'_i \geq 0$, go to 4.
   b. If m is less than or equal to the number of demand nodes and $S'_i = 0$, set $x_{[m]i} = 0$ for all values of m from its current value to the number of demand nodes and stop.
   c. If m is greater than the number of demand nodes, stop.

Once all of the V_i values have been computed (given by equations 6.12 to 6.14), we can find values for the location variables y_i in the Lagrangean problem by solving the problem described below:

$$min \sum_{i=1}^{m} (F_i + V_i) y_i \tag{6.16}$$

s.t.

$$\sum_{i=1}^{m} S_i y_i \geq \sum_{j=1}^{n} D_j \tag{6.17}$$

$$y_i = 0,1 \quad \forall\, i \in m \tag{6.18}$$

The objective function (6.16) states that the contribution of candidate facility i to the Lagrangean objective function will be equal to the fixed cost of locating at candidate facility i plus the assignment variables if we locate at candidate facility i. If we relax the integrality constraints (6.18) and replace them with $0 \leq y_i \leq 1 \;\; \forall\, i \in m,$ we could solve the optimization problem (6.16 to 6.18) with

the same greedy algorithm used for problem (6.12-6.14). However, since this optimization problem is quite simple, we chose to solve it using the CPLEX 12.6 solver, obtaining the optimal value for the location decision variables and insuring the facilities are opened, as we will see in the Projection Method.
After we get the values for the decision variables x_{ij} we are ready to calculate the remaining subgradient parameters. The agility parameter π is initialized with the value of 2, and every three consecutive failures to improve the lower bound, it is divided by 2 in order to decrease the step size (Δ). The agility parameter is restarted every 30 iterations. The step size (Δ) is calculated as follows:

$$\Delta = \frac{\pi(Z_{UB} - Z(\lambda))}{||\delta||} \tag{6.19}$$

In the equation (6.19), Z_{UB} is an upper bound of the original problem and $Z(\lambda)$ is the best lower bound found so far. The *j-th* component of the subgradient (δ) is:

$$\delta_j = 1 - \sum_{i=1}^{m} x_{ij} \tag{6.20}$$

Finally, to determine the set of λ multipliers that maximizes the Lagrangean function $Z(\lambda)$, the subgradient method requires the generation of new sequences of multipliers, one for each iteration of the Lagrangean Relaxation:

$$\lambda_{j+1} = \lambda_j + \Delta\delta_j \tag{6.21}$$

6.3.2. Projection Method

Once we obtain the lower bound, $Z(\lambda)$, then the dual solution is projected to the primal solutions space by solving the Transportation Problem (TP) of the decision variables $y_i = 1$ for any $i \in I$, and adding to the TP's objective function value the fixed costs of the selected facilities i, thus obtaining the upper bound (Z_{UB}). If we get an infeasible solution, then a simple procedure makes the solution feasible by opening sufficient facilities to serve all customers. This is

accomplished by selecting facilities by descending order of setup cost and opening facilities until all demand is met. For solving the TP, we used the commercial solver CPLEX 12.6 to obtain the optimal solution for this integer programming problem and retrieve the values of the assignment variables x_{ij}.

6.3.3. Primal Method

After the Projection Method obtains a primal feasible solution, the Dual-RAMP algorithm uses it as the initial solution for a simple improvement method based on tabu search that uses the classical ADD/DROP [14] local search neighborhood structures.

The ADD and DROP neighborhood structures are used one at a time and a move can only be applied if it is not tabu (the type of move, add or drop, and the location of the facility is stored in the tabu list). The choice for selecting a facility for ADD and/or DROP is based on choosing a facility by decreasing (DROP case) or increasing (ADD case) order of the total sum allocation for all customers. This choice will penalize facilities that contribute to the allocation cost growth and favor facilities that could make a reduction on this cost. If all moves are tabu then the first item in the tabu list is removed to allow new moves. Solving many TP is very expensive in terms of computational time, so we chose to apply the first improvement criteria, which means that we apply a ADD or DROP move whenever it improves the current solution in a given neighborhood.

Improvement Method

1. Start with solution S (from the Projection Method) and set $S_{best} = S$
2. Let C be the candidate list of facilities and V_SC the objective function value for adding or dropping a facility C' of C
3. Let $V_{Best} = V_SC$, where V_{Best} is the objective function value of S_{Best}
4. Set $noImprove = 0$, $tabuList = 0$;
5. **Do**
6. $ADD(checkTabu(\text{C}'))$
7. **If** $V_{Best} > V_SC$ **then**
8. $noImprove = 0$
9. $V_{Best} = V_SC$
10. $S_{best} = S$
11. **End if**

12. $DROP(checkTabu(\text{C}'))$
13. **If** $V_{Best} > V_SC$ **then**
14. $noImprove = 0$
15. $V_{Best} = V_SC$
16. $S_{best} = S$
17. **End if**
18. **If** $tabuList > maxTabuList$ **then**
19. $deleteHeadTabuItem()$
20. $tabuList --$
21. **End if**
22. $noImprove ++$
23. $UpdateTabuList()$
24. **While** $(\boldsymbol{noImprove \leq maxIter})$

The algorithm alternates the search between the dual side and the primal side until one of the four stopping criteria is achieved:

1 The agility parameter is less than 0.005;
2 The norm reaches the value 0;
3 The maximum number of iterations is reached;
4 The difference between the upper bound (Z_{UB}) and lower bound ($Z(\lambda)$) is less than 1.

6.4. Computational Results

The performance of the proposed Dual-RAMP algorithm was evaluated on a standard set of benchmark instances. The first one is the well-known OR-Library data set (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html) proposed by Beasley [11]. This set has 49 instances with known optimal solutions, which have the following sizes (facilities x customers): 16x50, 25x50 and 50x50 for the small instances and 100x1000 for the large ones. The second set of instances (TBED1) was presented by Avella and Boccia [5] (http://www.ing.unisannio.it/boccia) and contains 100 instances also with known optimal solutions. In this set the small instances have the following sizes: 300x300, 300x1500, 500x500 and 700x700. The large instances have 1000x1000 facilities and customers. The last set of instances (TESTBED A, B and C) contains the larger ones. Avella *et al.* [6] proposed TESTBED A and B (http://wpage.unina.it/sforza/test) and TESTBED C was introduced by Guastaroba and Speranza [60] (http://www-c.eco.unibs.it/~guastaro/InstancesCFLP.html). This

set is composed by 800x4400, 1000x1000, 1000x4000, 1200x3000 and 2000x2000 instances (445 in total), with different costs structures and different capacities. To scale these capacities the authors used a ratio with the following values: 1.1, 1.5, 2, 3, 5 and 10. Each of the five subsets shown in Table 4 contains six groups (of five instances), each with a specific ratio (for example, in TESTBEDA, 5 out of the 30 instances belonging to the 800x4400 subset have a ratio of 1.1, the next 5 have a ratio of 1.5, and so on). For these large instances (TESTBED A, B and C) no optimal solution is known (or it is not proven yet). Table 4 summarizes the data sets considered in our computational experiments, where instances belonging to the same data set are divided into subsets according to their size.

Data Set	N. Instances	N. Facilities	N. Customers
ORLIB			
1	13	16	50
2	12	25	50
3	12	50	50
4	12	100	1000
TBED1			
1	20	300	300
2	20	300	1500
3	20	500	500
4	20	700	700
5	20	1000	1000
TESTBED A/B/C			
1	30 (TESTBED B = 25)	800	4400
2	30	1000	1000
3	30	1000	4000
4	30	1200	3000
5	30	2000	2000

Table 4: Data sets used to evaluate the Dual-RAMP algorithm concerning the CFLP.

The algorithm was coded in C programming language and run on an Intel Pentium I7 2.40 GHz with 8GB RAM under Ubuntu operating system (only one processor was used) using CPLEX 12.61 to solve the Transportation Problem. To compare our Dual-RAMP algorithm with the state-of-the-art approaches for the CFLP we show in Tables 5-12 the results reported for these algorithms on the data sets previously described. The comparison between algorithms needs different tables, since some authors do not provide results for all instances.

The Dual-RAMP algorithm was compared with Guastaroba and Speranza's [60] Kernel Search (KS) procedure, Sun's [120] Tabu Search (TS) algorithm, Beasley's [9] Lagrangean Relaxation heuristic (LR), the Hybrid Approach (HA - Bee Algorithm with Mixed Integer Programming) proposed by Cabrera *et al.* [81], the Branch-and-Cut-and-Price algorithm proposed by Avella & Boccia [5] (AB), the Lagrangean Relaxation proposed by Avella *et al.* [6] (LR-A) and Rahmani & Mirhassani's [98] Firefly and Genetic Algorithm (F-GA).

Table 5 displays the results for the OR-Library instances divided by small, large and a subset of the large instances (to allow the comparison with a specific algorithm). In Tables 2 and 3, the "Gap" column was computed as $\frac{(UB - Z^*)}{UB} * 100$ (Z^* is the optimal solution) and the "CPU" column shows the computational time (in seconds) needed to achieve UB (the upper bound).

ORLIB	KS		TS		LR		HA		AB		**Dual-RAMP**	
	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
Small	0.00	0.63	0.00	0.24	0.02	1.49	-	-	-	-	**0.00**	**10.18**
Large	0.00	2158.83	0.07	48.63	0.21	75.79	-	-	0.00	415.79	**0.00**	**215.64**
Average	0.00	1079.73	0.04	24.44	0.12	38.64	-	-	-	-	**0.00**	**112.91**
CAPA8000	0.00	3604.88	-	-	0.24	73.65	0.00	367.74	-	-	**0.00**	**85.43**
CAPB8000	0.00	1999.73	-	-	0.40	131.84	0.00	317.47	0.00	319.26	**0.00**	**99.92**
CAPC5000	0.00	2621.94	-	-	0.00	105.74	0.00	129.34	0.00	577.30	**0.00**	**210.85**

Table 5: Results table for the ORLIB instances.

Our algorithm found the optimal solution for all the 49 ORLIB instances, outperforming the other five algorithms. Although computational times comparison is difficult when the algorithms run on different machines, it is apparent that our Dual-RAMP algorithm is exceptionally efficient on these types of instances. In particular, in the large instances subset, the Dual-RAMP obtained all optimal solutions in less than 216 seconds, which is very fast considering the size of these problems. In order to allow the comparison with the Bee algorithm (HA), we show the results for instances CAPA8000, CAPAB8000 and CAPAC5000. Although HA and KS achieved all optimal solutions, they used much more computational time (in average) than the Dual-RAMP.

Table 6 shows the results for the TBED1 data set. For two instances (1000x1000-11 and 1000x1000-12) the optimality has not been proven, so we used the best upper bound reported in [5].

	KS		AB		**Dual-RAMP**	
TBED1	Gap	CPU	Gap	CPU	**Gap**	**CPU**
300x300	0.00	57.82	0.00	327.99	**0.02**	**96.52**
300x1500	0.00	68.68	0.00	807.07	**0.05**	**674.82**
500x500	0.00	225.66	0.00	1518.56	**0.06**	**286.10**
700x700	0.00	795.80	0.00	6569.55	**0.06**	**709.15**
1000x1000	0.00	1745.87	0.00	46895.14	**0.22**	**627.67**
Average	0.00	578.77	0.00	11223.66	**0.08**	**478.85**

Table 6: Results table for the TBED1 instances.

For the TBED1 instances, our approach did not achieve all optimal solutions but it produced near optimal solutions in reduced computational times. KS and AB obtained better average percentage deviations from the optimal solution (or the best upper bound reported in [5]) than Dual-RAMP, but AB needed much higher computational times.

Table 7 displays the results for the instances from TESTBED A, B and C. Since the optimal solution is not known, we compute the "Gap" column as $\frac{(UB - LB)}{UB} * 100$. The "CPU" column continues to show the computational time (in seconds) needed to achieve UB. The LB values were obtained with our Lagrangean Relaxation, because we found very good lower bounds. Since the original instances for the TESTBEDC are not available, we used the instances provided by the KS authors, that where generated as described in Avella *et al*. [6]. Since the instances are not the same, we do not show the results for the LR-A algorithm on TESTBEDC.

	KS		LR-A		**Dual-RAMP**	
TESTBEDA	Gap	CPU	Gap	CPU	**Gap**	**CPU**
1000x1000	0.10	336.64	0.57	75.37	**0.23**	**152.66**
1000x4000	0.27	1539.77	0.67	304.45	**0.46**	**1714.60**
1200x3000	0.18	1570.96	0.63	150.43	**0.32**	**1336.42**

	KS		LR-A		**Dual-RAMP**	
TESTBEDA	Gap	CPU	Gap	CPU	**Gap**	**CPU**
2000x2000	0.07	1382.68	0.51	165.40	**0.24**	**964.11**
800x4400	0.33	1349.87	0.71	209.27	**0.31**	**1610.40**
Average	0.19	1235.99	0.62	180.98	**0.31**	**1155.64**
TESTBEDB						
1000x1000	0.34	1409.52	0.60	48.03	**0.79**	**477.24**
1000x4000	0.34	1519.93	1.21	129.85	**1.04**	**4270.93**
1200x3000	0.36	1727.18	0.78	108.78	**0.74**	**2956.66**
2000x2000	0.40	2073.38	0.96	161.25	**1.12**	**2588.78**
800x4400	0.33	1497.19	1.98	113.22	**0.90**	**4034.01**
Average	0.35	1645.44	1.10	112.23	**0.92**	**2865.53**
TESTBEDC						
1000x1000	3.57	1358.44	-	-	**0.91**	**2069.96**
1000x4000	2.09	465.63	-	-	**0.17**	**7934.40**
1200x3000	2.94	1001.21	-	-	**0.28**	**7481.75**
2000x2000	4.65	1833.15	-	-	**1.17**	**11391.92**
800x4400	1.51	265.86	-	-	**0.09**	**6121.58**
Average	2.95	984.86	-	-	**0.52**	**6999.92**

Table 7: Results table for the TESTBED A, B and C instances.

For these very large instances, the Dual-RAMP algorithm shows its robustness by being very consistent with the previous results, since it achieved good quality solutions in reasonable computational times. Dual-RAMP produced very competitive results for TESTBED A and B and proved to be very effective for TESTBEDC despite needing much higher computational times. Dual-RAMP achieved a 0.52% average Gap, a much better value than the 2.95% obtained by the KS approach.

Table 8 allows the comparison between our algorithm and the Rahmani & Mirhassani's [98] Firefly and Genetic Algorithm (F-GA). These authors do not report results for TESTBEDC and they also do not provide the computational times. Additionally, for TESTBED A and B they only show results for the instances with ratios 2, 5 and 10.

	F-GA		**Dual-RAMP**	
TESTBEDA	Gap	CPU	**Gap**	**CPU**
1000x1000	0.48	-	**0.35**	**184.23**
1000x4000	0.47	-	**0.69**	**1983.14**
1200x3000	0.42	-	**0.41**	**1603.88**
2000x2000	0.39	-	**0.35**	**931.06**
800x4400	0.48	-	**0.41**	**1932.41**
Average	0.45	-	**0.44**	**1326.94**
TESTBEDB				
1000x1000	0.49	-	**1.09**	**552.73**
1000x4000	0.96	-	**0.92**	**4010.70**
1200x3000	0.62	-	**0.60**	**2999.71**
2000x2000	0.58	-	**1.67**	**2892.97**
800x4400	1.11	-	**1.06**	**3454.50**
Average	0.75	-	**1.07**	**2782.12**

Table 8: Comparison with F-GA (TESTBED A and B).

For TESTBEDA, Dual-RAMP achieved better average quality solutions than F-GA. Only for the 1000x4000 instances F-GA outperformed Dual-RAMP. For TESTBEDB, the proposed algorithm produced a better average Gap for three (800x4400, 1000x4000 and 1200x3000) out of the five datasets, but the overall average Gap is worse than F-GA's.

Table 9 shows the overall results for TESTBEDA identifying the ratio for every group of instances (each group has 5 instances). "Average 1" displays the average values for all instances in each subset. Since F-GA does not report results for all ratio values, "Average 2" allows the comparison with F-GA, because it only considers the groups of instances for which this algorithm presents results.

TESTBEDA									
Size	Ratio	LR-A		KS		F-GA		**Dual-RAMP**	
		Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
1000x1000	1.1	0.78	102.80	0.00	244.96	-	-	**0.03**	**62.43**
1000x1000	1.5	0.68	93.50	0.00	439.37	-	-	**0.03**	**174.62**
1000x1000	2	0.77	93.40	0.01	174.06	0.61	-	**0.01**	**97.08**

TESTBEDA

Size	Ratio	LR-A		KS		F-GA		**Dual-RAMP**	
		Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
1000x1000	3	0.36	77.30	0.05	203.57	-	-	**0.26**	**126.26**
1000x1000	5	0.37	46.70	0.16	402.84	0.48	-	**0.26**	**103.60**
1000x1000	10	0.43	38.50	0.37	555.04	0.36	-	**0.76**	**352.00**
Average 1		0.57	75.37	0.10	336.64	-	-	**0.23**	**152.66**
Average 2		0.52	59.53	0.18	377.31	0.48	-	**0.35**	**184.23**
2000x2000	1.1	0.65	145.10	0.00	692.70	-	-	**0.04**	**616.07**
2000x2000	1.5	0.97	140.30	0.00	1419.26	-	-	**0.05**	**1538.03**
2000x2000	2	0.72	139.50	0.00	635.45	0.57	-	**0.16**	**1235.55**
2000x2000	3	0.20	286.90	0.02	1146.45	-	-	**0.27**	**837.40**
2000x2000	5	0.19	153.10	0.10	1736.04	0.26	-	**0.20**	**531.17**
2000x2000	10	0.35	127.50	0.28	2666.17	0.33	-	**0.70**	**1026.46**
Average 1		0.51	165.40	0.07	1382.68	-	-	**0.24**	**964.11**
Average 2		0.42	140.03	0.13	1679.22	0.39	-	**0.35**	**931.06**
1200x3000	1.1	0.78	180.10	0.01	516.51	-	-	**0.06**	**857.04**
1200x3000	1.5	0.62	151.70	0.01	690.99	-	-	**0.16**	**1325.28**
1200x3000	2	0.64	122.40	0.01	726.31	0.22	-	**0.09**	**854.93**
1200x3000	3	0.45	176.40	0.14	1806.03	-	-	**0.45**	**1024.53**
1200x3000	5	0.41	91.30	0.19	2612.25	0.38	-	**0.32**	**1500.57**
1200x3000	10	0.90	180.70	0.73	3073.70	0.66	-	**0.83**	**2456.15**
Average 1		0.63	150.43	0.18	1570.96	-	-	**0.32**	**1336.42**
Average 2		0.65	131.47	0.31	2137.42	0.42	-	**0.41**	**1603.88**
1000x4000	1.1	0.82	180.10	0.00	821.77	-	-	**0.38**	**1394.28**
1000x4000	1.5	0.54	151.90	0.02	791.28	-	-	**0.17**	**1750.07**
1000x4000	2	0.56	802.40	0.04	1020.75	0.58	-	**0.25**	**1367.06**
1000x4000	3	0.51	161.10	0.28	1972.92	-	-	**0.15**	**1193.87**
1000x4000	5	0.81	146.00	0.59	2448.83	0.41	-	**0.89**	**1500.95**
1000x4000	10	0.80	385.20	0.67	2183.09	0.43	-	**0.93**	**3081.39**
Average 1		0.67	304.45	0.27	1539.77	-	-	**0.46**	**1714.60**
Average 2		0.72	444.53	0.43	1884.22	0.47	-	**0.69**	**1983.13**
800x4400	1.1	0.77	137.10	0.01	685.96	-	-	**0.10**	**954.76**
800x4400	1.5	0.80	128.70	0.03	555.99	-	-	**0.16**	**1481.28**
800x4400	2	0.57	398.90	0.06	730.22	0.43	-	**0.24**	**1288.21**
800x4400	3	0.76	131.50	0.52	2149.16	-	-	**0.38**	**1429.15**

TESTBEDA									
Size	Ratio	LR-A		KS		F-GA		**Dual-RAMP**	
		Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
800x4400	5	0.84	196.70	0.60	3118.79	0.55	-	**0.77**	**1998.55**
800x4400	10	0.50	262.70	0.78	859.11	0.47	-	**0.23**	**2510.48**
Average 1		0.71	209.27	0.33	1349.87	-	-	**0.31**	**1610.40**
Average 2		0.64	286.10	0.48	1569.38	0.48	-	**0.41**	**1932.41**
Overall Average 1		0.62	180.98	0.19	1235.99	-	-	**0.31**	**1155.64**
Overall Average 2		0.59	212.33	0.31	1529.51	0.45	-	**0.44**	**1326.94**

Table 9: Overall results for TESTBEDA.

Our approach achieved an overall average Gap of 0.31%, obtaining very consistent results for all subsets. Dual-RAMP clearly outperforms LR-A and F-GA but only manages to obtain better average Gap results than KS in one subset (800x4400).

Table 10 displays the overall results for TESTBEDB considering the six previously defined ratios, except for ratio 10 in subset 800x4400, where no instances are available in literature.

TESTBEDB									
Size	Ratio	LR-A		KS		F-GA		**Dual-RAMP**	
		Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
1000x1000	1.1	0.65	46.60	0.01	306.88	-	-	**0.17**	**397.99**
1000x1000	1.5	0.34	131.10	0.03	657.87	-	-	**0.34**	**369.17**
1000x1000	2	0.43	46.20	0.13	1731.27	0.61	-	**0.39**	**379.18**
1000x1000	3	0.87	31.40	0.88	2129.64	-	-	**0.97**	**438.10**
1000x1000	5	0.79	20.90	0.73	2742.11	0.42	-	**1.90**	**587.63**
1000x1000	10	0.49	12.00	0.29	703.22	0.44	-	**0.96**	**691.38**
Average 1		0.60	48.03	0.34	1378.50	-	-	**0.79**	**477.24**
Average 2		0.57	26.37	0.38	1725.53	0.49	-	**1.08**	**552.73**
2000x2000	1.1	0.92	153.50	0.00	360.74	-	-	**0.47**	**2642.53**
2000x2000	1.5	1.70	135.70	0.02	1426.69	-	-	**0.42**	**1848.72**
2000x2000	2	0.61	347.30	0.06	2497.92	0.55	-	**0.55**	**1952.35**
2000x2000	3	0.81	112.90	0.73	2264.33	-	-	**0.83**	**2362.54**
2000x2000	5	0.89	125.60	0.93	2963.12	0.47	-	**2.60**	**2522.71**

TESTBEDB									
Size	Ratio	LR-A		KS		F-GA		**Dual-RAMP**	
		Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
2000x2000	10	0.81	92.50	0.65	2927.49	0.72	-	**1.87**	**4203.84**
Average 1		0.96	161.25	0.40	2073.38	-	-	**1.12**	**2588.78**
Average 2		0.77	188.47	0.55	2796.18	0.58	-	**1.67**	**2892.97**
1200x3000	1.1	1.31	107.10	0.01	683.68	-	-	**0.56**	**2821.72**
1200x3000	1.5	0.45	251.60	0.14	1939.52	-	-	**0.56**	**3148.40**
1200x3000	2	0.75	114.20	0.47	2636.51	0.36	-	**0.37**	**2212.87**
1200x3000	3	0.85	76.60	0.80	2764.22	-	-	**1.53**	**2770.74**
1200x3000	5	0.62	48.40	0.48	1691.98	0.59	-	**0.72**	**3388.45**
1200x3000	10	0.67	54.80	0.23	647.18	0.91	-	**0.72**	**3397.82**
Average 1		0.78	108.78	0.36	1727.18	-	-	**0.74**	**2956.66**
Average 2		0.68	72.47	0.39	1658.56	0.62	-	**0.60**	**2999.71**
1000x4000	1.1	2.58	131.90	0.04	789.04	-	-	**0.97**	**5404.14**
1000x4000	1.5	0.40	176.90	0.23	1963.62	-	-	**0.44**	**4064.33**
1000x4000	2	0.65	113.90	0.45	2037.48	0.76	-	**0.86**	**3884.84**
1000x4000	3	0.86	111.20	0.56	2044.55	-	-	**2.06**	**4124.98**
1000x4000	5	0.75	105.70	0.28	1563.30	0.23	-	**1.35**	**4282.03**
1000x4000	10	2.02	139.50	0.68	664.07	1.88	-	**0.55**	**3865.24**
Average 1		1.21	129.85	0.37	1510.34	-	-	**1.04**	**4270.93**
Average 2		1.14	119.70	0.47	1421.62	0.96	-	**0.92**	**4010.70**
800x4400	1.1	3.56	127.80	0.05	450.60	-	-	**0.52**	**5538.80**
800x4400	1.5	0.54	161.30	0.28	2021.79	-	-	**0.52**	**4048.39**
800x4400	2	0.73	114.10	0.51	2014.33	0.71	-	**1.19**	**3515.53**
800x4400	3	0.81	105.40	0.49	2030.54	-	-	**1.23**	**3612.84**
800x4400	5	2.42	89.90	0.32	968.67	1.13	-	**1.06**	**3454.50**
800x4400	10	3.79	80.80	-	-	1.09	-	**-**	**-**
Average 1		1.61	119.70	0.33	1497.19	-	-	**0.90**	**4034.01**
Average 2		3.11	85.35	0.32	968.67	1.11	-	**1.06**	**3454.50**
Overall Average 1		1.10	112.23	0.36	1637.32	-	-	**0.92**	**2865.53**
Overall Average 2		1.25	98.47	0.42	1714.11	0.75	-	**1.07**	**2782.12**

Table 10: Overall results for TESTBEDB.

Dual-RAMP was outperformed by the other algorithms in the first to subsets (1000x1000 and 2000x2000), but still managed to produce better results than

LR-A in some ratio groups. In the other three subsets, our algorithm provided competitive results, obtaining better solution quality average results than LR-A and F-GA.

Finally, in Table 11 we can see the overall results for TESTBEDC.

TESTBEDC

Size	Ratio	KS		**Dual-RAMP**	
		Gap	CPU	**Gap**	**CPU**
1000x1000	1.1	0.290	1606.525	**1.277**	**1627.282**
1000x1000	1.5	1.992	1855.660	**1.600**	**2830.381**
1000x1000	2	3.560	1852.993	**1.319**	**2451.702**
1000x1000	3	3.813	1850.644	**0.798**	**2356.854**
1000x1000	5	5.226	906.415	**0.343**	**1832.797**
1000x1000	10	6.557	78.371	**0.102**	**1320.752**
	Average	3.573	1358.435	**0.906**	**2069.961**
2000x2000	1.1	0.356	2045.588	**1.273**	**10641.146**
2000x2000	1.5	2.241	2031.086	**1.762**	**13040.536**
2000x2000	2	4.569	2022.590	**2.084**	**12657.531**
2000x2000	3	4.962	2015.882	**1.109**	**14242.666**
2000x2000	5	7.042	2008.304	**0.530**	**10598.207**
2000x2000	10	8.720	875.455	**0.232**	**7171.438**
	Average	4.648	1833.151	**1.165**	**11391.921**
1200x3000	1.1	0.384	1048.181	**0.594**	**9640.190**
1200x3000	1.5	2.036	2027.960	**0.475**	**8891.180**
1200x3000	2	3.489	2023.192	**0.350**	**7140.738**
1200x3000	3	3.058	425.350	**0.171**	**7318.431**
1200x3000	5	4.039	252.989	**0.059**	**6191.035**
1200x3000	10	4.659	229.573	**0.024**	**5708.945**
	Average	2.944	1001.208	**0.279**	**7481.753**
1000x4000	1.1	0.309	348.611	**0.632**	**14559.013**
1000x4000	1.5	1.623	1209.205	**0.171**	**7829.817**
1000x4000	2	2.618	402.107	**0.135**	**7581.392**
1000x4000	3	2.147	282.039	**0.043**	**7118.837**
1000x4000	5	2.693	278.641	**0.023**	**5777.780**
1000x4000	10	3.177	273.153	**0.009**	**4739.561**
	Average	2.095	465.626	**0.169**	**7934.400**

TESTBEDC					
Size	Ratio	KS		**Dual-RAMP**	
		Gap	CPU	**Gap**	**CPU**
800x4400	1.1	0.237	283.104	**0.373**	**10463.241**
800x4400	1.5	1.242	296.990	**0.095**	**7087.461**
800x4400	2	1.936	266.327	**0.061**	**5998.334**
800x4400	3	1.536	252.196	**0.025**	**5430.097**
800x4400	5	1.895	248.608	**0.006**	**4331.471**
800x4400	10	2.235	247.953	**0.002**	**3418.865**
Average		1.513	265.863	**0.094**	**6121.578**
Overall Average		2.955	984.856	**0.523**	**6999.923**

Table 11: Overall results for TESTBEDC.

For TESTBEDC, the Dual-RAMP approach reached excellent results in terms of solution quality, beating by far the KS algorithm in every subset, but with much higher computational times. KS only managed to get better results in the groups with 1.1 ratio.

In Table 12 we introduce new best-known solutions that Dual-RAMP managed to find for specific instances in TESTBED A. The best-known upper bounds are provided by Avella *et al.* [6] and we also show the solutions reported by KS for the same instances. Although KS succeeded in improving some of the best-known solutions, Dual-RAMP found even better ones.

TESTBEDA							
Instances	Best-Known Upper Bound	KS			**Dual-RAMP**		
		UB	Gap	CPU	UB	Gap	CPU
800x4400-10	443271.8	441232.75	-0.46	431.67	**439959.86**	**-0.75**	**1394.35**
800x4400-30	58042.1	58309.09	0.46	410.67	**58041.44**	**-0.001**	**1790.85**
1000x1000-3	931344.1	931344.10	0.00	305.45	**931305.42**	**-0.004**	**96.69**
1000x1000-5	915766	915674.42	-0.01	333.39	**915650.79**	**-0.01**	**57.80**
1000x1000-13	331688	331157.30	-0.16	103.60	**329553.10**	**-0.64**	**44.13**
1000x1000-14	335263.9	334626.90	-0.19	417.23	**333648.10**	**-0.48**	**94.05**
1000x4000-17	116557.1	116300.67	-0.22	1101.36	**116065.75**	**-0.42**	**1067.43**
1000x4000-19	114194	113840.00	-0.31	3165.20	**113826.58**	**-0.32**	**1338.22**
1200x3000-11	413265.1	410950.82	-0.56	1011.15	**410105.32**	**-0.76**	**741.24**

TESTBEDA							
Instances	Best-Known Upper Bound	KS			**Dual-RAMP**		
		UB	Gap	CPU	UB	Gap	CPU
2000x2000-21	93282.4	93235.76	-0.05	1143.79	**93095.59**	**-0.20**	**467.55**

Table 12: New best-known solutions for some specific instances in TESTBEDA.

In summary, our Dual-RAMP algorithm proved to be a robust approach for the solution of the CFLP, by effectively solving the best-known (small, large and very large) instances available in the literature.

6.5. Conclusions

This paper describes a Dual-RAMP algorithm for the CFLP that competes with the best-known algorithms for the solution of this problem. Despite the fact that only the first level of sophistication of the RAMP method was implemented, the proposed algorithm managed to produce excellent results for the complete testbed in reasonable computational times, and even introduced ten new best-known solutions. We conjecture that our algorithm owes its advantage to the premise that a judicious exploration of primal-dual relationships provides an effective interplay between intensification and diversification that is absent in search methods confined to the primal solution space. The consistent encouraging results obtained by RAMP applications to hard combinatorial optimization problems certainly invite further studies in the application of the method. In particular, the impressive results reported by the present study strongly suggest extending our RAMP approach to the solution of other facility location problems.

7. Dual-RAMP for the Capacitated Single Allocation Hub Location Problem

Abstract: In this paper we address the Capacitated Single Allocation Hub Location Problem (CSAHLP) in which the objective is to choose the optimal set of hubs from all nodes in a given network so that the total cost of allocating all nodes to the chosen hubs is minimum. We propose a Dual-RAMP algorithm that takes advantage of the primal-dual relationships by combining Lagrangean subgradient search with an improvement method and obtaining information from both primal and dual solutions spaces. We present an extensive computational study that proves the effectiveness of our approach for the solution of the CSAHLP that includes seven new best-known solutions.

Keywords: RAMP, CSAHLP, Hub Location, Adaptive Memory Programming, Lagrangean Relaxation.

7.1. Introduction

The Capacitated Single Allocation Hub Location Problem (CSAHLP) is a well-known combinatorial optimization problem that belongs to the class of the NP-Hard problems [94]. The model used in this work was given by Contreras *et al.* [23] and is described as follows. Consider the complete graph $G = (N, A)$, where N is the set of nodes $N = \{1,2,\dots,n\}$, that correspond to origins/destinations as well as potential hub locations. Let w_{ij} be the flow between i and j, $O_i = \sum_{j \in N} w_{ij}$ the outgoing flow from node $i \in N$, and $D = \sum_{i \in N} O_i$ the total flow generated in the graph. For each node $i \in N$, let b_i denote the capacity and f_i the fixed set-up cost of hub i. The capacity of a hub represents an upper bound on the total incoming flow that can be processed in the hub. Thus, it refers to the sum of the flow generated at the nodes that are assigned to the hub. The distance between nodes i and j is assumed to satisfy the triangle inequality, and is denoted by d_{ij}. We will use these distances as a measure of the per unit flow transportation costs along the links of the graph. These distances are weighted by some discount factors, denoted by χ, α and δ, to represent the collection, transfer and distribution costs per unit of flow,

respectively. Being single allocation implies that each node can only be assigned to one hub. The objective consists in choosing the set of nodes to be established as hubs, that minimizes the total cost of assigning all the non-hub nodes to the chosen hubs, without violating the capacity constraint of the hubs. The total cost of routing the flow along the path $i-j-k-m$ (these are the paths between origin destination pairs, where i and j represents the origin and destination, respectively, and k and m are the hubs to which i and j are allocated, respectively) is given by:

$$F_{ijkm} = w_{ij}(\chi d_{ik} + \alpha d_{km} + \delta d_{mj}) \tag{7.1}$$

and for each pair $i,\ k \in N$ the following sets of binary decision variables are defined:

$$Z_{ik} = \begin{cases} 1\ if\ node\ i\ is\ assigned\ to\ hub\ k; \\ 0\ otherwise. \end{cases} \tag{7.2}$$

Variable Z_{kk} denotes the establishment (or not) of a hub at node k, when $i = k$. An additional set of binary variables is defined to indicate if there is flow through each link of the graph. For each $i,j,k,m \in N$ the existence of flow is defined by:

$$X_{ijkm} = \begin{cases} 1\ if\ flow\ from\ i\ to\ j\ goes\ via\ hubs\ k\ and\ m; \\ 0\ otherwise. \end{cases} \tag{7.3}$$

The mathematical formulation for the CSAHLP is:

$$min \sum_{k\in N} f_k Z_{kk} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N} F_{ijkm} X_{ijkm} \tag{7.4}$$

s.t.

$$\sum_{k\in N}\sum_{m\in N} X_{ijkm} = 1\ \forall\ i,j \in N \tag{7.5}$$

$$Z_{ik} \leq Z_{kk}\ \forall\ i,k \in N \tag{7.6}$$

$$\sum_{m\in N} X_{ijkm} = Z_{ik}\ \forall\ i,j,k \in N \tag{7.7}$$

$$\sum_{k \in N} X_{ijkm} = Z_{jm} \; \forall \; i, j, m \in N \tag{7.8}$$

$$\sum_{i \in N} O_i z_{ik} \leq b_k Z_{kk} \; \forall \; k \in N \tag{7.9}$$

$$\sum_{k \in N} b_k Z_{kk} \geq D \tag{7.10}$$

$$Z_{ik} \in \{0,1\} \; \forall \; i, k \in N \tag{7.11}$$

$$X_{ijkm} \in \{0,1\} \; \forall \; i, j, k, m \in N \tag{7.12}$$

Constraints (7.5) assure that every single node is assigned to one hub, whereas constraints (7.6) guarantee that no non-hub node is assigned to a node that is not an hub. Constraints (7.7) insure that if node i is assigned to hub k then all the flow from this node to any other (fixed) node j must go through some other hub m. Constraints (7.8) state a similar interpretation regarding to the flow sent to a node j, assigned to hub m, from some node i. Constraints (7.9) guarantee that the total incoming flow of nodes assigned to a hub does not overdo its capacity. (7.10) concerns the demand constraint and satisfying this aggregated (demand) constraint is necessary to assure the feasibility of the Z variables. This can also be achieved by adding up all constraints (7.9), and taking into consideration equalities (7.5) and (7.7). Constraint (7.10) is redundant for the formulation but we include it in order to apply the Lagrangean Relaxation, as referred in [23].

7.2. Related Work

The Hub Location Problem (HLP) has been extensively studied over the past years (as described in several surveys [4,19,41,93]), resulting in a great number of different algorithmic approaches. The most common applications for the HLP could initially be found in telecommunication, logistics or airline companies, but nowadays many other industries take advantage of the hub location model to reduce transportation costs. The HLP has many variants, with single or multiple allocation, for uncapacitated or capacitated hub location. The capacitated version of this problem is not as studied as the uncapacitated. The

same happens with the type of allocation, since the single variant has been getting much more attention than the multiple. The first linear programming formulations for tackling these problems were proposed by Campbell [20].

We can find in the literature numerous exact and heuristic methods for the CSAHLP. With regard to the exact approaches, a Branch & Bound [80] technique is found in the work of Ernst and Krishnamoorthy [36], in which the authors presented a modified problem formulation of the CSApHLP (where p is the number of necessary hubs to be opened) to solve the CSAHLP. Correia *et al*. [27] explored the problem formulation, leading to the inclusion of additional sets of constraints that helped to decrease the computational time needed to solve the problem to optimality for small instances. Labbé *et al*. [75] examined the polyhedral properties and produced a Branch & Cut algorithm for this problem. Costa *et al*. [28] presented a second objective function to the model with an interactive procedure to generate non-dominated solutions. In this work, some computational results are shown with a comparison between single and double objective functions. Almeida *et al*. [2] proposed a Local Branching method that is based on Branch & Cut with some heuristic and local search techniques to explore neighborhood solutions. Stanojević and Marić [114] produced an algorithm to solve the CSAHLP and the USAHLP (the uncapacitated version) with exact and heuristic methods. The main idea is to search the entire hub configuration and, for each configuration, an exact approach (the authors used the commercial solver CPLEX) is used to solve the allocation subproblem. They have tested the algorithm on small and large instances obtaining very good results but with expensive computational times.

Being NP-Hard, the time needed to solve this problem is very costly. Due to this fact, several heuristic procedures are presented in the literature. Besides the mentioned Branch & Bound approach proposed by Ernst and Krishnamoorthy [36], they also presented a random descent and a simulated annealing algorithm for this problem. Population based heuristics can also be found in the literature, in the works of Stanimirović [115], Mohammad [89], Almeida *et al*. [3] and Baker [7] using Genetic Algorithms, and using Ant Colony optimization in Stützle and Dorigo [32] and in Randall [99]. Chen [21] presented an heuristic procedure that assigned three levels to the problem. The first level determines the number of

hubs required; the second level chooses the location of the hubs, given the number of hubs found in first level; and the third level determines the allocation for the hubs defined in the second level. With this approach, Chen achieved very good results for small and large instances with reasonable computational times. Contreras *et al*. [23] introduced a Lagrangean Relaxation (LR) and reduction tests based on LR bounds that reduced the size of the problem and consequently the computational time. More recently, Marić [84] presented a Variable Neighborhood Search [88] for choosing the set of hubs and its locations. The author also used a commercial solver to solve the allocation part. The reported computational results refer only to small instances. Finally, Stenojević *et al*. [117] proposed a hybrid procedure combining a parallel Branch & Bound technique with an evolutionary approach to solve this problem. This method proved to be very effective for small and large instances, with very competitive results in terms of solution quality and even more efficient in terms of computational time, due to parallel computing.

7.3. RAMP Algorithm for the CSAHLP

Proposed by Rego [100] in 2005, the Relaxation Adaptive Memory Programming (RAMP) metaheuristic framework combines fundamental principles of mathematical relaxation with concepts of adaptive memory programming techniques, covering primal and dual solutions spaces, with the objective of incorporating information obtained by both sides of the problem.

The RAMP method allows different levels of sophistication, depending on the intensification desired for the primal or dual search. At the first level of sophistication (Dual-RAMP), this framework explores more intensively the dual side, restricting the primal side interaction to the projection of dual solutions to the primal solutions space and to the improvement of these solutions. Higher levels of sophistication (PD-RAMP) allow a more intensive exploration of the primal side, incorporating the simple level, the Dual-RAMP, with more complex memory structures.

Several combinatorial optimization problems have already been solved by RAMP applications, producing excellent results, in some cases with new best-known solutions. Some examples of RAMP approaches with different levels of

sophistication are the uncapacitated facility location problem [45], the linear ordering problem [45], the resource constrained project scheduling problem [106], among others complex combinatorial optimization problems.

We propose a Dual-RAMP algorithm for the CSAHLP that uses a Lagrangean Relaxation [23] on the dual side and an Improvement Method on the primal side. With this simple level of sophistication, the algorithm explores and combines dual and primal solutions spaces, achieving very good results for small, medium and large instances.

7.3.1. Dual Method

The Dual Method proposed for this algorithm relies on the exploration of the dual solutions space with subgradient optimization to solve the dual problem obtained by the Lagrangean Relaxation. At each iteration of the subgradient optimization, a solution for the relaxed problem is obtained, then a Projection Method projects it to the primal solutions space and an Improvement Method tries to improve it.

Specifically, if we relax constraints (7.7) and (7.8), we obtain the following optimization problem:

$$L(u,v) = min \sum_{k\in N} f_k Z_{kk} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N} F_{ijkm} X_{ijkm} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N} u_{ijk}\left(\sum_{m\in N} X_{ijkm} - Z_{ik}\right) + \sum_{i\in N}\sum_{j\in N}\sum_{m\in N} v_{ijm}\left(\sum_{m\in N} X_{ijkm} - Z_{jm}\right) \quad (7.13)$$

s.t.

(7.5), (7.6), (7.9), (7.10), (7.11) and (7.12)

The remaining problem can be separated in two subproblems, denoted as $L_Z(u,v)$ and $L_X(u,v)$, for Z space variables and for X space variables, respectively.

7.3.1.1. Z space variables

For Z space variables, decomposing the main problem $L(u,v)$ with respect to these variables, will produce the following subproblem:

$$L_Z(u,v) = min \sum_{k\in N} f_k Z_{kk} - \sum_{i\in N}\sum_{k\in N}\left(\sum_{j\in N}(u_{ijk}+v_{jik})\right)Z_{ik} \tag{7.14}$$

s.t.

$$Z_{ik} \leq Z_{kk}\ \forall\ i,k \in N \tag{7.15}$$

$$\sum_{i\in N} O_i z_{ik} \leq b_k Z_{kk}\ \forall\ k \in N \tag{7.16}$$

$$\sum_{k\in N} b_k Z_{kk} \geq D \tag{7.17}$$

$$Z_{ik} \in \{0,1\}\ \forall\ i,k \in N \tag{7.18}$$

The solution $L_Z(u,v)$ is given by a set of hubs to be opened $(Z_{kk}=1)$ and by the allocation of nodes to these hubs $(Z_{ik}=1, i \neq k)$. In this subproblem, and as mentioned in [23], a feasible solution does not require nodes to be assigned to just one open hub. So, it can happen that a node is not assigned to any hub, or that it is assigned to more than one open hub. If a hub is in $k \in N$, so that $Z_{kk}=1$, the remaining nodes that will be assigned to hub k can be identified by solving the following binary knapsack problem:

$$KP_k = min \sum_{i\neq k}\left(\sum_{j}(u_{ijk}+v_{jik})\right)Z_{ik} \tag{7.19}$$

s.t.

$$\sum_{i\neq k} O_i z_{ik} \leq (b_k - O_k)\ \forall\ k \in N \tag{7.20}$$

$$Z_{ik} \in \{0,1\}\ \forall\ i,k \in N, i \neq k \tag{7.21}$$

Each KP_k is a binary knapsack problem that assesses the maximum gain of opening a hub at node k with respect to the coefficients of gain $\sum_{j\in N} -(u_{ijk} + v_{jik})$. Constraints (7.20) represent the capacity of hub k (when it is open) that is accessible for the rest of the nodes. The optimal set of hubs in $L_Z(u,v)$ can be obtained by solving the problem:

$$L_Z(u,v) = min \sum_{k\in N} \left(f_k - \sum_{j\in N} (u_{kjk} + v_{jkk}) - KP_k \right) Z_{kk} \tag{7.22}$$

s.t.

$$\sum_{k\in N} b_k Z_{kk} \geq D \tag{7.23}$$

$$Z_{kk} \in \{0,1\} \; \forall \; k \in N \tag{7.24}$$

Consequently, we can obtain the solution for $L_Z(u,v)$ by solving a series of |N| + 1 knapsack problems. For this we follow the same strategy as Contreras *et al.* [23] and use the algorithm of Martello, Pisinger and Toth [85].

7.3.1.2. X space variables

For X space variables, decomposing the main problem $L(u,v)$ in respect to these variables, will produce the following subproblem:

$$L_X(u,v) = min \sum_{i\in N} \sum_{j\in N} \sum_{k\in N} \sum_{m\in N} (F_{ijkm} + u_{ijk} + v_{ijm}) X_{ijkm} \tag{7.25}$$

s.t.

$$\sum_{k\in N} \sum_{m\in N} X_{ijkm} = 1 \; \forall \; i,j \in N \tag{7.26}$$

$$X_{ijkm} \in \{0,1\} \; \forall \; i,j,k,m \in N \tag{7.27}$$

It is trivial to note that $L_X(u,v)$ is a semi-assignment problem. Thus, for each pair (i,j), it can be decomposed into $(N-1)^2$ independent semi-assignment problems of the form:

$$SAP_{ij} = min \sum_{k \in N} \sum_{m \in N} \left(F_{ijkm} + u_{ijk} + v_{ijm}\right) X_{ijkm} \tag{7.28}$$

s.t.

$$\sum_{k \in N} \sum_{m \in N} X_{ijkm} = 1 \tag{7.29}$$

$$X_{ijkm} \in \{0,1\} \; \forall \; k, m \in N \tag{7.30}$$

For a given pair (i, j), the SAP_{ij} can be solved by the following rule:

$$X_{ijkl} = 1 \qquad \text{for } F_{ijkl} = min\{F_{ijkm} + u_{ijk} + v_{ijm} | k, m \in N\} \tag{7.31}$$

$$X_{ijkm} = 0 \qquad \text{Otherwise} \tag{7.32}$$

The relaxed problem $L(u, v)$ can be obtained by the sum of the two subproblems, one for each space variables, as previously shown:

$$L(u, v) = L_Z(u, v) + L_X(u, v) \tag{7.33}$$

The lower bound computation, Z_D will be:

$$Z_D = max_{u,v} L(u, v) \tag{7.34}$$

7.3.2. Subgradient optimization

For solving the Lagrangean dual problem we used subgradient optimization. At each iteration, the solution for the relaxed problem $L(u, v)$ is obtained, which is projected to the primal solutions space by a Projection Method and then an Improvement Method tries to improve it.

We start with the lower bound $Z_D = 0$, an initial value for the initial upper bound Z_{UB} (that we will see in the Primal Method) and the Lagrangean multipliers $\lambda_{u,v} = 0$. The agility parameter π is initialized with the value 2, it is

divided by 2 every 25 consecutive iterations without improving Z_D and it is reset to the original value every 100 iterations in order to decrease the step size (Δ). The step size (Δ) is calculated as follows:

$$\Delta = \frac{\pi(Z_{UB} - L(u,v))}{||\delta||} \tag{7.35}$$

For a given vector (u, v), let $z(u, v)$ and $x(u, v)$ represent the optimal solution of $L(u, v)$. Then, a subgradient $\delta(u, v)$ of $L(u, v)$, is given by the following expression:

$$\delta(u,v) = \left(\left(\sum_m X_{ijkm}(u) - Z_{ik} \right)_{i,j,k}, \left(\sum_k X_{ijkm}(u) - Z_{jm} \right)_{i,j,m} \right) \tag{7.36}$$

Finally, to determine the set of multipliers, $\lambda_{u,v}$, that maximizes the Lagrangean function $L(u, v)$, the subgradient method requires generating new sequences of multipliers, one for each iteration of the Lagrangean Relaxation:

$$\lambda_{u,v}^{iter+1} = \lambda_{u,v} + \Delta\delta(u,v) \tag{7.37}$$

7.3.3. Primal Method

The Primal Method starts after the Projection Method projects the dual solution onto the primal solutions space. For X solutions space we have to solve the $|N| + 1$ knapsack problems (equations 7.19-7.24) and for Z solutions space we solve the semi-assignment problems associated with each pair (i, j) (equations 7.28-7.32).

At each iteration, the Dual-RAMP algorithm starts the primal exploration with an Improvement Method that uses the primal feasible solution that we got from the Dual Method as the initial solution. Often a feasible solution is not produced by the Dual Method iteration, so we use a simple method to transform an infeasible solution into a feasible one. This method is divided into two stages.

In the first stage, the Projection Method tries to open hubs. If it fails to open at least one hub, then it chooses to open the hub with the highest supply (and all the nodes are assigned to it). The second stage assumes that the Projection Method opened at least one hub, and verifies if the opened hubs can supply all nodes assigned to them. If not, it opens hubs until the nodes demand is met without violating the hubs total supply. This second stage is accomplished by the following procedure:

Turn solution feasible Procedure

1. Let $HList$ be the set of hubs opened by the Projection Method
2. Let $DemandList[h]$ be the set of all nodes allocated to hub $h \in HList$ ordered by ascending order of demand
3. **for** every $h \in HList$ **do**
4. **while** $(\boldsymbol{h_{supply} < 0})$
5. deallocate tail node $n \in DemandList[h]_{tail}$
6. let $CostList[n]$ be the set of all nodes ordered by ascending order of cost regarding node n
7. **for** every $m \in CostList[n]$ do
8. **If** $m \in HList$ and $m_{capacity} \geq n_{demand}$ **then**
9. Include n in $DemandList[m]$
10. **end if**
11. **end for**
12. **if** n cannot be allocated to one of $HList$
13. **set** $n \in HList$
14. **end if**
15. Update supply and demand for all hubs and nodes
16. **end while**
17. **end for**

The Dual-RAMP algorithm proceeds with the Primal Method by trying to improve the feasible primal solution with a simple Improvement Method. This method has three neighborhood structures: shift, swap and close, that we can describe as follows:

- Shift (nodes): the shift procedure shifts nodes from one open hub to another, until no more improvement can be made to the objective function. Basically (for every node), the procedure tries to improve the objective function by shifting a node from one open hub to another;

- Swap (nodes): the swap procedure swaps nodes from open hubs until no more improvement can be made to the objective function. For every node, this procedure analysis the benefit of swapping two nodes assigned to two different hubs and does the swapping if it improves the objective function;
- Swap (hubs): the swap procedure swaps open hubs until no more improvement can be made to the objective function. For every hub, this procedure analysis the gain of swapping two open hubs and does the swapping (by reassigning all the nodes from one open hub to the other) if it improves the objective function;
- Close (hubs): the close procedure looks for profit in closing an open hub and reassigning its nodes to the remaining open hubs based on minimum distances.

The Improvement Method performs the procedures (Shift-nodes, Swap-nodes, Swap-hubs and Close-hubs), one at a time, until no improvement to the objective function is possible. The algorithm continues alternating between the dual and the primal solutions spaces until one of the four stopping criteria is achieved:

1. The agility parameter is less than 0.005;
2. The norm reaches the value 0;
3. The maximum number of iterations is reached;
4. The difference between the upper bound (Z_{UB}) and lower bound (Z_D) is less than 1.

7.4. Computational Results

The performance of the Dual-RAMP algorithm was evaluated on a standard AP (Australia Post) dataset, composed by 56 instances, introduced by Ernst and Krishnamoorthy [36] and obtained in Beasley's OR-Library [11]. These instances result from the mail flows in an Australian city and contain up to 200 nodes. The flow is not symmetric, that is, $W_{ij} \neq W_{ji}$ and the collection, transfer and distribution ratios are $\chi = 3, \alpha = 0.75$ and $\delta = 2$.

These instances are divided in small, medium and large sizes and have two different types of setup costs and hubs capacities. We have tight (T) setup costs when these costs increase with the amount of flow generated in the node and we have loose (L) setup costs when the instances do not have this characteristic. The tight setup costs are supposed to be more difficult to solve. For the hubs capacities, we have the same representation, but in this case related to how tightly (T) or loosely (L) constrained the instances are. For more information about the instances, please refer to Ernst and Krishnamoorthy [36] or Contreras *et al*. [23]. Each group has four instances corresponding to the four possible combinations of setup costs and capacities (LL, LT, TL or TT). For instances up to 50 nodes the optimal solution is known. For all the other instances, we used the best-known solutions reported in Contreras *et al*. [23].
The algorithm was coded in C programming language and run on an Intel Pentium I7 2.40 GHz (only one processor was used) with 8GB RAM under Ubuntu operating system. The Dual-RAMP algorithm was compared with the state of the art algorithms for the solution of the CSAHLP. All results tables show the average percent deviation from the optimal/best-known solution (Gap) and the associated computational time (CPU) in seconds. The best-known approaches for the CSAHLP are a Genetic Algorithm (GA) [115], an exact approach (LB - Local Branching) [2], a Variable Neighborhood Search (VNS) [84] and an Evolutionary Algorithm and parallel Branch-and-Bound approach (EA) [117]. We cannot compare our computational times with the EA approach since our algorithm is not parallel and we only use one processor. Nevertheless, we include the solutions obtained with this approach in the results tables. For the GA and the EA algorithms, the average results were achieved in a specific number of runs with different parameters. Our algorithm obtained the reported solutions with a single run.
Table 13 summarizes the results for small, medium and large AP data instances grouped by size. LB and VNS do not provide results for all AP datasets. In all results tables the value of the "Gap" column was computed as $(UB - Z^*)/UB * 100$ (Z^* is the optimal/best-known solution and UB is the obtained upper bound) and the "CPU" column shows the computational time (in seconds) needed to achieve UB.

AP	GA		LB		VNS		EA		**Dual-RAMP**	
	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
10	0.07	1.02	0.00	1.56	0.05	0.17	0.00	0.09	**0.00**	**0.03**
20	0.09	2.33	0.00	4.45	0.07	1.67	0.00	0.31	**0.00**	**0.93**
25	1.21	3.03	0.00	9.11	0.10	7.71	0.12	0.57	**0.00**	**2.52**
40	1.97	5.77	0.00	1054.88	1.01	58.68	0.00	0.88	**0.00**	**19.22**
50	1.07	8.96	0.04	15694.55	2.08	245.37	0.21	1.03	**0.12**	**54.30**
Average	0.88	4.22	0.01	3352.91	0.66	62.72	0.07	0.58	**0.02**	**15.40**
100	2.36	70.07	-	-	-	-	0.48	2.80	**0.01**	**635.68**
200	1.75	397.09	-	-	-	-	0.91	27.71	**0.01**	**8815.52**

Table 13: Aggregated results for the AP data instances.

Table 13 clearly shows that the Dual-RAMP approach achieved excellent quality results for all small instances. Only the LB algorithm reached a better average Gap for the 50 nodes group, due to one specific instance, as we will see in the detailed results table. However, the LB approach needs a higher computational effort to achieve such solutions. For medium and large instances (up to 100 and 200 nodes, respectively), our algorithm outperforms all other algorithms (GA and EA) in solution quality (managing to find a new best-known solution, as we will see in the following tables) but with higher computational times.

Table 14 to Table 16 detail the computational results for small, medium and large instances. LB and VNS do not provide results for instances of medium and large sizes. GA and EA show complete results for small instances, but for medium and large the authors only report results for instances with 100 and 200 nodes.

AP (small)	LR		GA		LB		VNS		EA		**Dual-RAMP**	
	OPT	CPU	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
10LL	224250.05	0.03	0.13	0.94	0.00	2.04	0.00	0.16	0.00	0.09	**0.00**	**0.02**
10LT	250992.26	0.03	0.00	1.11	0.00	1.63	0.00	0.42	0.00	0.10	**0.00**	**0.03**
10TL	263399.94	0.06	0.12	1.07	0.00	1.03	0.00	0.07	0.00	0.10	**0.00**	**0.04**
10TT	263399.94	0.05	0.05	0.97	0.00	1.52	0.18	0.03	0.00	0.08	**0.00**	**0.03**
20LL	234690.96	0.38	0.00	2.17	0.00	3.33	0.00	0.91	0.00	0.15	**0.00**	**0.13**

AP (small)	LR		GA		LB		VNS		EA		**Dual-RAMP**	
	OPT	CPU	Gap	CPU	Gap	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
20LT	253517.4	1.64	0.07	2.41	0.00	5.39	0.00	0.79	0.00	0.53	**0.00**	**1.78**
20TL	271128.18	0.19	0.00	2.57	0.00	2.85	0.26	1.82	0.00	0.17	**0.00**	**0.11**
20TT	296035.4	0.16	0.30	2.16	0.00	6.22	0.00	3.16	0.00	0.41	**0.00**	**1.71**
25LL	238977.95	1.97	0.78	3.07	0.00	7.56	0.00	6.06	0.00	0.16	**0.00**	**1.44**
25LT	276372.5	2.55	0.98	2.97	0.00	10.48	0.20	6.31	0.20	1.37	**0.00**	**3.91**
25TL	310317.64	1.05	2.25	3.08	0.00	6.46	0.00	7.62	0.00	0.16	**0.00**	**0.79**
25TT	348369.15	2.47	0.85	3.00	0.00	11.93	0.21	10.84	0.28	0.61	**0.00**	**3.95**
40LL	241955.71	7.24	0.08	5.27	0.00	29.63	0.00	32.85	0.00	0.43	**0.00**	**21.51**
40LT	272218.32	6.81	3.54	6.46	0.00	70.12	1.67	68.13	0.00	1.77	**0.00**	**20.58**
40TL	298919.01	1.59	0.00	5.62	0.00	22.93	0.00	38.86	0.00	0.21	**0.00**	**11.16**
40TT	354874.1	11.39	4.25	5.75	0.00	4096.85	2.37	94.86	0.00	1.12	**0.00**	**23.63**
50LL	238520.59	13.00	0.27	7.45	0.00	55.19	0.36	89.79	0.00	0.26	**0.00**	**45.25**
50LT	272897.49	63.98	0.52	8.53	0.00	512.00	1.83	171.54	0.53	1.80	**0.00**	**57.96**
50TL	319015.77	20.02	0.72	8.87	0.00	92.32	0.48	82.40	0.00	0.24	**0.00**	**52.02**
50TT	417440.99	77.16	2.79	10.98	0.16	62118.70	5.63	637.74	0.33	1.84	**0.48**	**61.95**
Average		10.59	0.88	4.22	0.01	3352.91	0.66	62.72	0.07	0.58	**0.02**	**15.40**

Table 14: AP results – small instances.

For the small instances (whose optimal solution is known), we can see from Table 14 that our algorithm achieved the optimal solution in all instances except for the 50TT instance, for which Dual-RAMP obtained a 0.48% deviation from the optimal solution. This is the most difficult instance of the small ones, as demonstrated by the results produced by all algorithms, since none attained the optimal solution. Nevertheless, the proposed algorithm achieved almost every optimal solution in extremely reduced computational times, under 16 seconds in average (note that a direct comparison is not possible because the algorithms run on different machines).

Table 15 shows the results for the medium instances (from 60 to 100 nodes) whose optimal solution is not known (the "Best-Known" column displays the best-known solutions reported by Contreras *et al.* [23]). "Average 1" specifies

the average for all instances and "Average 2" only considers the 100 nodes instances.

AP (medium)	LR		GA		EA		**Dual-RAMP**	
	Best-Known	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
60LL	225917.21	21.56	-	-	-	-	**0.00**	**96.35**
60LT	253761.98	94.80	-	-	-	-	**-0.06**	**131.99**
60TL	252496.66	2.45	-	-	-	-	**0.00**	**80.53**
60TT	351274.72	33.39	-	-	-	-	**-0.02**	**116.59**
70LL	236817.35	145.05	-	-	-	-	**0.00**	**210.88**
70LT	257454.36	185.70	-	-	-	-	**-0.20**	**242.59**
70TL	271283.82	15.50	-	-	-	-	**0.00**	**191.83**
70TT	387380.2	48.99	-	-	-	-	**0.01**	**218.72**
75LL	238024.22	145.44	-	-	-	-	**0.00**	**275.69**
75LT	256188.12	203.43	-	-	-	-	**0.00**	**293.74**
75TL	303363.55	44.83	-	-	-	-	**0.00**	**249.57**
75TT	347189.81	85.06	-	-	-	-	**0.00**	**283.37**
90LL	224195.72	237.95	-	-	-	-	**0.00**	**552.75**
90LT	246026.24	515.84	-	-	-	-	**0.01**	**639.75**
90TL	281561.56	110.39	-	-	-	-	**0.10**	**535.89**
90TT	337008.93	141.34	-	-	-	-	**0.00**	**586.25**
100LL	246713.97	873.80	1.07	56.05	0.96	1.22	**0.00**	**612.47**
100LT	256155.33	1039.60	3.19	81.97	0.65	4.04	**0.00**	**694.01**
100TL	362950.09	397.02	0.77	66.29	0.10	0.88	**0.05**	**558.54**
100TT	474186.73	347.87	4.39	75.96	0.20	5.04	**-0.03**	**677.68**
	Average 1	234.50	-	-	-	-	**-0.01**	**362.46**
	Average 2	664.57	2.36	70.07	0.48	2.80	**0.01**	**635.68**

Table 15: AP results – medium instances.

For this group of instances, we can see that the computational time increases. Our approach outperformed the other algorithms (GA and EA) in terms of solution quality, obtaining a 0.01% average Gap for the larger (and most difficult to solve of the medium size group) instances in reasonable computational times. Comparing with LR, the proposed algorithm achieved equal or better solutions for all instances. Dual-RAMP provided an improvement on the best-known solution for instances 60LT, 60TT, 70LT and 100TT, obtaining an overall average Gap of -0.01%.

In Table 16 we present the results for the large instances (from 125 to 200 nodes) whose optimal solution is not known. "Average 1" specifies the average for all instances and "Average 2" only considers the 200 nodes instances.

AP (large)	LR		GA		EA		**Dual-RAMP**	
	Best-Known	CPU	Gap	CPU	Gap	CPU	**Gap**	**CPU**
125LL	239920.75	2731.23	-	-	-	-	**0.00**	**1632.28**
125LT	251259.16	2096.22	-	-	-	-	**0.00**	**2002.9**
125TL	246486.69	89.11	-	-	-	-	**0.00**	**1467.18**
125TT	291807.35	287.50	-	-	-	-	**0.01**	**1735.87**
150LL	234765.44	6671.82	-	-	-	-	**0.00**	**3079.33**
150LT	250186.53	5852.48	-	-	-	-	**-0.07**	**3787.68**
150TL	263356.18	1009.87	-	-	-	-	**-0.31**	**2892.01**
150TT	323992.37	1802.13	-	-	-	-	**-0.08**	**3725.75**
175LL	277997.58	10352.05	-	-	-	-	**0.00**	**5929.32**
175LT	251540.8	13172.84	-	-	-	-	**0.00**	**7212.21**
175TL	244860.41	1019.50	-	-	-	-	**0.00**	**5317.71**
175TT	308310.13	1856.37	-	-	-	-	**1.36**	**6864.23**
200LL	231069.5	19590.75	0.70	424.52	0.77	6.76	**0.00**	**8123.18**
200LT	268820.57	23212.58	1.86	410.41	1.42	72.92	**-0.42**	**10344.32**
200TL	273443.81	2801.48	3.63	325.88	0.45	5.81	**0.00**	**7362.94**
200TT	290841.84	3111.90	0.81	427.57	1.02	25.33	**0.47**	**9431.64**
	Average 1	5978.61	-	-	-	-	**0.06**	**5056.78**
	Average 2	12179.18	1.75	397.09	0.91	27.71	**0.01**	**8815.52**

Table 16: AP results – large instances.

For these instances, considered the harder ones, we can see that Dual-RAMP achieves the same quality level as with smaller instances, demonstrating that our algorithm is the best approach for this problem. Comparing with GA and EA, our approach beat all in terms of solution quality, achieving a 0.01% average deviation from the best-known solutions. Our algorithm also produced equal or better results than LR, except for instances 125TT, 175TT and 200TT. Dual-RAMP found new best-known solutions for instances 150LT, 150TL, 150TT and 200LT.

Table 17 shows the improvement that Dual-RAMP and EA algorithms obtained on the best-known solutions reported by Contreras *et al.* [23]. For the EA

approach, these improvements cannot be seen in the previous tables because, as explained previously, the EA results are averages of a specific number (20) of runs. In Table 17, we report the results obtained in individual runs that allowed EA to improve some best-known solutions. The "New Solution" column displays the upper bound obtained by the Dual-RAMP algorithm.

AP	LR	EA	**Dual-RAMP**		
	Best-Known	Gap	**New Solution**	**Gap**	**CPU**
60LT	253761.98	-	**253616.48**	**-0.06**	**131.99**
60TT	351274.72	-	**252496.63**	**-0.02**	**116.59**
70LT	257454.36	-	**256936.95**	**-0.20**	**242.59**
100TT	474186.73	-0.02	**474068.91**	**-0.03**	**677.68**
150LT	250186.53	-	**250002.59**	**-0.07**	**3787.68**
150TL	263356.18	-	**262543.09**	**-0.31**	**2892.01**
150TT	323992.37	-	**323721.56**	**-0.08**	**3725.75**
200LT	268820.57	**-0.59**	267702.59	-0.42	**7362.94**
200TT	290841.84	**-0.03**	-	0.47	10344.32

Table 17: New best-known solutions.

EA improved the best-known solutions for instances 100TT, 200LT and 200TT but Dual-RAMP managed to find an even better solution for the instance 100TT. Our algorithm also improved the best-known solution for the 200LT instance in 0.42% but, in this case, EA achieved a higher improvement of 0.59%. For the 200TT, we obtained a 0.47% deviation from the best-known and EA managed to improve it in 0.03%. Overall, Dual-RAMP found seven new best-known solutions and EA only two.

Taking in consideration that the Dual-RAMP algorithm is sequential, we believe that combining parallel computing with the RAMP framework will produce even better results than the ones presented in this paper.

7.5. Conclusions

In this study, we propose a Dual-RAMP algorithm for the CSAHLP that combines a Lagrangean Relaxation approach with a local search approach based on three classic neighborhood structures: shift, swap and close.

The results presented (that include seven new best-known solutions), clearly show the advantage of the Dual-RAMP algorithm over the best algorithms for the CSAHLP currently in the literature.

The quality of the results seems to stem from the appropriate interconnection of approaches capable of effectively exploiting both dual and primal problems. The combination of the information collected in the exploration of the dual solutions space (which treats constraints) with the information obtained with the exploration of the primal solutions space (that treats solutions) enriches the set of available solutions, generating good results in reasonable computational times.

The robustness of the proposed Dual-RAMP algorithm, demonstrated on a standard dataset of small, medium and large instances, suggests that this approach can be applied to other difficult combinatorial optimization problems (and with relevant applications), with the same degree of success.

8. RAMP algorithms for the Capacitated Single Allocation p-Hub Location Problem

Abstract: We propose two Relaxation Adaptive Memory Programming (RAMP) algorithms for the Capacitated Single Allocation $\boldsymbol{p}$-Hub Location Problem (CSA$\boldsymbol{p}$HLP) for which the goal is to determine the set of $\boldsymbol{p}$ hubs in a network that minimizes the total cost of allocating all the non-hub nodes to the $\boldsymbol{p}$ hubs. The algorithms explore primal-dual relationships by appropriately combining "restricted" with "relaxation" procedures under the RAMP framework. The first algorithm, Dual-RAMP, combines Lagrangean Relaxation and Subgradient Optimization (in the dual side) with a simple Improvement Method (in the primal side). The second algorithm, PD-RAMP, incorporates Dual-RAMP with a Scatter Search procedure to create a Primal-Dual RAMP approach. The quality of the results carried out on a standard testbed shows that the RAMP approach clearly outperforms the state-of-the-art algorithms for the CSA$\boldsymbol{p}$HLP.

Keywords: RAMP, CSApHLP, Hub Location, Adaptive Memory Programming, Lagrangean Relaxation.

8.1. Introduction

The Capacitated Single Allocation p-Hub Location Problem (CSApHLP) is a vastly studied combinatorial optimization problem that belongs to the class of the NP-Hard problems [94]. In this study we used the formulation given by Contreras *et al.* [23] (considering the fixed set of required hubs) that can be described as follows.

Let $G = (N, A)$ be a complete graph where $N = \{1, 2, \dots, n\}$ is a set of nodes that represent origins/destinations and can be defined as hubs. Let w_{ij} be the flow between i and j, $O_i = \sum_{j \in N} w_{ij}$ the outgoing flow from node $i \in N$, and $D = \sum_{i \in N} O_i$ the total flow generated in the graph. For each node $i \in N$, let b_i denote the capacity and f_i the fixed set-up cost of hub i. The capacity of a hub represents an upper bound on the total incoming flow that can be processed in the hub. Thus, it refers to the sum of the flow generated at the nodes that are assigned to the hub. The distance between nodes i and j is assumed to satisfy the triangle inequality, and is denoted by d_{ij}. These distances will be used as a

measure of the per unit flow transportation costs along the links of the graph. They are weighted by some discount factors, denoted by χ, α and δ, to represent the collection, transfer and distribution costs per unit of flow, respectively. Each node can only be assigned to one hub, since the problem defines the allocation as single. The goal is to choose a set of p nodes to define as hubs that minimizes the total cost of assigning all the other nodes to the selected p hubs, satisfying the hubs capacity constraints. The total cost of routing the flow along the path $i-j-k-m$ (these are the paths between origin destination pairs, where i and j represents the origin and destination, respectively, and k and m are the hubs to which i and j are allocated, respectively) is given by:

$$F_{ijkm} = w_{ij}(\chi d_{ik} + \alpha d_{km} + \delta d_{mj}) \tag{8.1}$$

and for each pair i, $k \in N$ we have the following sets of binary decision variables:

$$Z_{ik} = \begin{cases} 1 \; if \; node \; i \; is \; assigned \; to \; hub \; k; \\ 0 \; otherwise. \end{cases} \tag{8.2}$$

When $i = k$, variable Z_{kk} represents the establishment (or not) of a hub at node k. The Z variables will be referred to as location/allocation variables. We define an additional set of binary variables that represent the amount of flow through each link of the graph. For each $i,j,k,m \in N$ let the existence of flow be defined by:

$$X_{ijkm} = \begin{cases} 1 \; if \; flow \; from \; i \; to \; j \; goes \; via \; hubs \; k \; and \; m; \\ 0 \; otherwise. \end{cases} \tag{8.3}$$

The mathematical formulation for CSAHLP is:

$$min \sum_{k \in N} f_k Z_{kk} + \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{m \in N} F_{ijkm} X_{ijkm} \tag{8.4}$$

s.t.

$$\sum_{k \in N} \sum_{m \in N} X_{ijkm} = 1 \; \forall \; i,j \in N \tag{8.5}$$

$$\sum_{k \in N} Z_{kk} = p \tag{8.6}$$

$$Z_{ik} \leq Z_{kk} \; \forall \, i,k \in N \tag{8.7}$$

$$\sum_{m \in N} X_{ijkm} = Z_{ik} \; \forall \, i,j,k \in N \tag{8.8}$$

$$\sum_{k \in N} X_{ijkm} = Z_{jm} \; \forall \, i,j,m \in N \tag{8.9}$$

$$\sum_{i \in N} O_i z_{ik} \leq b_k Z_{kk} \; \forall \, k \in N \tag{8.10}$$

$$\sum_{k \in N} b_k Z_{kk} \geq D \tag{8.11}$$

$$Z_{ik} \in \{0,1\} \; \forall \, i,k \in N \tag{8.12}$$

$$X_{ijkm} \in \{0,1\} \; \forall \, i,j,k,m \in N \tag{8.13}$$

Constraints (8.5) guarantee that every node is assigned to one single hub. Constraint (8.6) specifies that exactly p nodes are chosen to act as hubs, whereas constraints (8.7) impose that no node is assigned to a node that is not a hub. Constraints (8.8) state that if node i is assigned to hub k then all the flow from node i to any other (fixed) node j must go through some other hub m. Constraints (8.9) have a similar interpretation for the flow arriving to a node j assigned to hub m from some node i. Constraints (8.10) ensure that the overall incoming flow of nodes assigned to a hub does not exceed its capacity. Constraint (8.11) is the aggregated demand constraint. Note that satisfying this constraint is a necessary condition for the feasibility of the location/allocation vectors, that can be derived by adding up all constraints (8.10) and considering equalities (8.5) and (8.8). Constraint (8.11) is redundant for the formulation, as mentioned by Contreras *et al.* [23], but the author chose to include it in the model in order to strengthen the relaxation.

8.2. Related Work

Hub Location Problems (HLP) have capture the attention of researchers over the past years as is attested by several surveys [4,41,93,95] and a vast number of algorithmic approaches. The HLP scientific interest is due to the complexity

and to the many applications in industry (aviation [16], public transportation [91], among others). The HLP have many variants with regard to allocation (single or multiple) and capacity (uncapacitated or capacitated).

For the uncapacitated single and multiple allocation p-hub median problems (USApHMP and UMApHMP, respectively), several articles can be found in the literature. O'kelly [94] presented this problem and introduced the first quadratic mathematical formulation. The author proposed two heuristics and reported computational results considering 2, 3 and 4 open hubs. Klincewicz [70] presented and compared several interchange heuristics. Campbell [18] introduced integer formulations for multiple and single p-hub median problems (uncapacitated versions) and also presented two heuristics to tackle these problems. Ernst [37,38] proposed exact and heuristic methods for solving both USApHMP and UMApHMP, and created new datasets for these problems.

There are some articles in the literature that deal with the capacitated version of this problem but they do not consider the fixed cost of opening a hub, making the problem simpler to solve. Perez *et al.* [97] introduced a hybrid GRASP-Path Relinking approach, in which the GRASP procedure [43] is used to construct the population of the Path Relinking. They presented results for standard datasets and solved instances up to 100 nodes with values of p between 2 and 20. Stanimirović [116] produced a Genetic Algorithm with excellent results for small, medium and large instances in the literature.

In this paper, we tackle the most difficult capacitated version of the HLP that considers the fixed cost to open a hub in a specific location.

CSApHLP (this problem may be referred as CSApHMP, depending on whether authors consider – or not – the fixed cost to open a hub) is a recent problem, so only two articles can be found addressing it. Lu and Ting [83] produced a Lagrangean Relaxation procedure with subgradient optimization. They divided the problem formulation into two subproblems, as in Contreras *et al.* [23], obtaining the Z space variables and the X space variables subproblems. They solved the Z space variables subproblem using the Gurobi solver and the assignment problems of the X space variables subproblem with a simple procedure that finds the shortest path between nodes. The authors presented computational results for the AP dataset and solved instances up to 50 nodes

with $p = 2, 3, 4, 5$. The newest article found addressing this problem is a reactive GRASP proposed by Ting *et al*. [122]. Computational results were reported on a standard dataset of instances with up to 50 nodes and for $p = 2, 3, 4, 5$. As far as we know, there are no results in the literature for instances with more than 50 nodes. Our algorithm produced results for instances up to 200 nodes (the largest in the literature) with $p = 2, 3, 4, 5$.

8.3. RAMP Algorithms for the CSApHLP

Relaxation Adaptive Memory Programming (RAMP) is a metaheuristic framework proposed by Rego [100] founded on the exploration of the primal-dual relationships of the optimization problem, guiding research with Adaptive Memory Programming (AMP) in a concept similar to the one introduced in Tabu search techniques. RAMP can be defined in general terms, as the combination of AMP concepts and metaheuristic techniques with principles of mathematical relaxation, covering the dual and primal solution spaces, with the objective of obtaining crucial information about the problem.

Originally, this method suggests combinations of the previously mentioned concepts involving different levels of sophistication. At the first level (Dual-RAMP) the method explores more intensively the dual side of the problem and at the second level (the PD-RAMP) the method integrates the Dual-RAMP approach with more advanced strategies to exploit primal-dual relationships more extensively.

Lagrangean Relaxation and surrogate constraints, among other relaxation techniques, are examples of dual strategies. For primal components, Scatter Search [77][52] or Path Relinking [52], are suggested as examples of methods suitable for adaptive memory usage.

The RAMP method can be described as an incremental process, beginning with the implementation of Dual-RAMP and successively move to more complex forms of the RAMP approach.

The method has proved extremely effective in finding optimal and near-optimal solutions for a variety of combinatorial optimization problems such as the capacitated minimum spanning tree [101], the linear ordering problem [45] and the resource constrained project scheduling problem [106], among others.

We propose two RAMP algorithms for the CSApHLP. We start with a Dual-RAMP approach that combines a Lagrangean Relaxation with a Subgradient Procedure on the dual side and an Improvement Method on the primal side. We proceed with a PD-RAMP approach that incorporates Dual-RAMP with an evolutionary procedure (Scatter Search) aiming to intensify the search in the primal side. With the effective combination of dual and primal solutions spaces, our approach manages to achieve excellent results for small, medium and large instances, demonstrating the advantages of the RAMP framework.

8.3.1. Dual Method

The algorithm uses Lagrangean Relaxation to explore the dual side of the problem based on the work of Contreras *et al.* [23] applied to the CSAHLP. The Dual Method explores the dual solutions space by solving the relaxed problem with subgradient optimization. At each iteration, a Projection Method is used to project the dual solution to the primal solutions space, followed by an Improvement Method that tries to improve the projected solution.

If we relax constraints (8.8) and (8.9), we obtain the following optimization problem:

$$L(u,v) = min \sum_{k\in N} f_k Z_{kk} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N} F_{ijkm} X_{ijkm} +$$
$$\sum_{i\in N}\sum_{j\in N}\sum_{k\in N} u_{ijk}\left(\sum_{m\in N} X_{ijkm} - Z_{ik}\right) + \qquad (8.14)$$
$$\sum_{i\in N}\sum_{j\in N}\sum_{m\in N} v_{ijm}\left(\sum_{m\in N} X_{ijkm} - Z_{jm}\right)$$

s.t.

(8.5), (8.6), (8.7), (8.10), (8.11), (8.12) and (8.13)

The remaining problem can be separated in two subproblems, denoted as $L_Z(u,v)$ and $L_X(u,v)$, for Z space variables and for X space variables, respectively.

Decomposing the main problem $L(u,v)$ in respect to the of Z space variables, results in the following subproblem:

$$L_Z(u,v) = min \sum_{k\in N} f_k Z_{kk} - \sum_{i\in N}\sum_{k\in N}\left(\sum_{j\in N}(u_{ijk} + v_{jik})\right) Z_{ik} \quad (8.15)$$

s.t.

$$\sum_{k\in N} Z_{kk} = p \quad (8.16)$$

$$Z_{ik} \leq Z_{kk} \; \forall \; i,k \in N \quad (8.17)$$

$$\sum_{i\in N} O_i z_{ik} \leq b_k Z_{kk} \; \forall \; k \in N \quad (8.18)$$

$$\sum_{k\in N} b_k Z_{kk} \geq D \quad (8.19)$$

$$Z_{ik} \in \{0,1\} \; \forall \; i,k \in N \quad (8.20)$$

A set of p hubs to be open $(Z_{kk} = p)$ and the allocation of the remaining nodes to the open hubs $(Z_{ik} = 1, i \neq k)$ constitute a solution for $L_Z(u,v)$. As mention by Contreras *et al*. [23] a feasible solution for this subproblem does not require that nodes be assigned to just one open hub. This means that we can have nodes that are not assigned to any hub, or nodes that are assigned to more than one open hub. If a hub is in $k \in N$, so that $Z_{kk} = 1$, the remaining nodes that will be assigned to hub k can be identified by solving the following binary knapsack problem:

$$KP_k = min \sum_{i\neq k}\left(\sum_{j}(u_{ijk} + v_{jik})\right) Z_{ik} \quad (8.21)$$

s.t.

$$\sum_{i\neq k} O_i z_{ik} \leq (b_k - O_k) \; \forall \; k \in N \quad (8.22)$$

$$Z_{ik} \in \{0,1\} \; \forall \; i,k \in N, i \neq k \quad (8.23)$$

Each KP_k is a binary knapsack problem that evaluates the maximum profit of opening a hub at node k with respect to the profit coefficients $\sum_j (u_{ijk} + v_{jik})$. Constraints (8.22) represent the capacity of hub k that is available for the rest of the nodes, when it is open.

The optimal set of hubs in $L_Z(u,v)$ can be obtained by solving the problem:

$$L_Z(u,v) = min \sum_{k \in N} \left(f_k - \sum_{j \in N} \left(u_{kjk} + v_{jkk} \right) - KP_k \right) Z_{kk} \tag{8.24}$$

s.t.

$$\sum_{k \in N} b_k Z_{kk} \geq D \tag{8.25}$$

$$Z_{kk} \in \{0,1\} \; \forall \; k \in N \tag{8.26}$$

This means that we can obtain the solution for $L_Z(u,v)$ by solving a series of |N| + 1 knapsack problems. To achieve this, we used the approach of Contreras *et al*. [23] and the algorithm proposed by Martello, Pisinger and Toth [85]. As it requires p hubs to be open as stated by constraint (8.16), and Contreras *et al*. Lagrangean Relaxation does not consider that, we force $L_Z(u,v)$ to open the required number of hubs. To select the hubs to be opened the nodes are ordered by decreasing value of the profit given by equations (8.21) to (8.23). Decomposing the main problem $L(u,v)$ in respect to the of X space variables, results in the following subproblem:

$$L_X(u,v) = min \sum_{i \in N} \sum_{j \in N} \sum_{k \in N} \sum_{m \in N} \left(F_{ijkm} + u_{ijk} + v_{ijm} \right) X_{ijkm} \tag{8.27}$$

s.t.

$$\sum_{k \in N} \sum_{m \in N} X_{ijkm} = 1 \; \forall \; i,j \in N \tag{8.28}$$

$$X_{ijkm} \in \{0,1\} \; \forall \; i,j,k,m \in N \tag{8.29}$$

$L_X(u,v)$ is evidently a semi-assignment problem, hence for each pair (i,j), it can be decomposed into $(N-1)^2$ independent semi-assignment problems of the form:

$$SAP_{ij} = min \sum_{k \in N} \sum_{m \in N} \left(F_{ijkm} + u_{ijk} + v_{ijm}\right) X_{ijkm} \tag{8.30}$$

s.t.

$$\sum_{k \in N} \sum_{m \in N} X_{ijkm} = 1 \tag{8.31}$$

$$X_{ijkm} \in \{0,1\} \; \forall \; k, m \in N \tag{8.32}$$

For each pair (i, j), the SAP_{ij} can be solved by using the following rule:

$$X_{ijwl} = 1 \qquad \text{for } F_{ijwl} = min\{F_{ijkm} + u_{ijk} + v_{ijm} | k, m \in N\} \tag{8.33}$$

$$X_{ijkm} = 0 \qquad \text{Otherwise} \tag{8.34}$$

The relaxed problem $L(u, v)$ can be obtained by the sum of the two subproblems, one for each space variables, as already shown:

$$L(u, v) = L_Z(u, v) + L_X(u, v) \tag{8.35}$$

The lower bound computation, Z_D will be:

$$Z_D = max_{u,v} L(u, v) \tag{8.36}$$

We used subgradient optimization to solve the Lagrangean dual problem. At each iteration, we obtain the solution for the relaxed problem $L(u, v)$, project it to the primal solutions space through a Projection Method followed by an Improvement Method.

The procedure starts with $Z_D = 0$ (the lower bound), an initial value for Z_{UB} (the upper bound that we will see in the Primal Method) and the Lagrangean multipliers $\lambda_{u,v} = 0$. The agility parameter π is initialized with the value 2, it is divided by 2 every 25 consecutive iterations without improving Z_D, and it is reset to the original value every 100 iterations in order to decrease the step size (Δ). The step size (Δ) is calculated as follows:

$$\Delta = \frac{\pi(Z_{UB} - L(u,v))}{||\delta||} \tag{8.37}$$

For a given vector (u,v), let $z(u,v)$, and $x(u,v)$ be the optimal solution to $L(u,v)$. Then, a subgradient $\delta(u,v)$ of $L(u,v)$, is given by:

$$\delta(u,v) = \left(\left(\sum_m X_{ijkm}(u) - Z_{ik} \right)_{i,j,k}, \left(\sum_m X_{ijkm}(u) - Z_{jm} \right)_{i,j,m} \right) \tag{8.38}$$

In order to determine the set of $\lambda_{u,v}$ multipliers that maximizes the Lagrangean function $L(u,v)$, the subgradient method needs to generate new sequences of multipliers, one for each iteration of the Lagrangean Relaxation:

$$\lambda_{u,v}^{iter+1} = \lambda_{u,v} + \Delta\delta(u,v) \tag{8.39}$$

8.3.2. Primal Method

The Dual-RAMP algorithm starts the primal exploration with an Improvement Method that tries to improve the primal feasible solution provided by the Dual Method. Sometimes a feasible solution is not reached at the end of the dual search, so we use a simple method to ensure feasibility. This method is divided in two stages. In the first stage, the Projection Method tries to open hubs. If it fails to open at least one hub, then it chooses to open the hub with the highest supply (and all the nodes are assigned to it). The second stage assumes that the Projection Method opens at least the required number of p hubs. If not, it opens hubs until the required number of p hubs are met. After that, the solution requires that all nodes to be allocated to open hubs without violating its capacity. This is accomplished by the following procedure:

Turn solution feasible Procedure

1. Let $HList$ be the set of hubs opened by the Projection Method
2. Let $DemandList[h]$ be the set of all nodes allocated to hub $h \in HList$ ordered by ascending order of demand
3. **for** every $h \in HList$ **do**
4. **while** $(\boldsymbol{h_{supply} < 0})$
5. deallocate tail node $n \in DemandList[h]_{tail}$
6. let $CostList[n]$ be the set of all nodes ordered by ascending order of cost regarding node n
7. **for** every $m \in CostList[n]$ do
8. **If** $m \in HList$ and $m_{capacity} \geq n_{demand}$ **then**
9. Include n in $DemandList[m]$
10. **end if**
11. **end for**
12. **if** n cannot be allocated to one of $HList$
13. **set** $n \in HList$
14. **end if**
15. Update supply and demand for all hubs and nodes
16. **end while**
17. **end for**

After assuring feasibility, the Dual-RAMP algorithm proceeds with the Improvement Method that considers the *shift*, *swap* and *close* classic neighborhood structures:

- Shift (nodes): the shift procedure shifts nodes from one open hub to another, until no more improvement can be made to the objective function. Basically (for every node), the procedure tries to improve the objective function by shifting a node from one open hub to another;
- Swap (nodes): the swap procedure swaps nodes from open hubs until no more improvement can be made to the objective function. For every node, this procedure analysis the benefit of swapping two nodes assigned to two different hubs and does the swapping if it improves the objective function;
- Swap (hubs): the swap procedure swaps open hubs until no more improvement can be made to the objective function. For every hub, this procedure analysis the gain of swapping two open hubs and does the

swapping (by reassigning all the nodes from one open hub to the other) if it improves the objective function;

- Close (hubs): The procedure (Turn Solution Feasible) that transforms a projected solution into a feasible one may open more hubs than necessary. As we need to open exactly p hubs, we verify, at each iteration, if the number of hubs differs from p and close a hub when necessary. Closing a hub involves looking for the best gain in closing an open hub (from the set of open hubs) and reassigning all the nodes to the remaining open hubs based on minimum distances.

The improvement method performs the procedures (Shift-nodes, Swap-nodes, Swap-hubs and Close-hubs), one at a time, until no improvement to the objective function is possible.

The Dual-RAMP algorithm continues alternating between the dual and the primal solutions spaces until one of the four stopping criteria is achieved:

1. The agility parameter is less than 0.005;
2. The norm reaches the value 0;
3. The maximum number of iterations is reached;
4. The difference between the upper bound (Z_{UB}) and lower bound (Z_D) is less than 1.

As described in Section 3, the RAMP framework can be implemented incrementally, starting with a straightforward Dual-RAMP approach and progressively evolving into more complex versions.

Although the simplest version of the framework (Dual-RAMP) produced extremely competitive results, we decided to implement a more sophisticated RAMP algorithm for the solution of the CSApHLP, to assess if it could produce even better results. This new approach (PD-RAMP) uses the same dualization and Improvement Method as the Dual-RAMP algorithm.

In PD-RAMP (Primal-Dual RAMP), advanced strategies are integrated, allowing a wider search of the primal solutions space. Primal and dual solutions are combined in an evolutionary way, using a common reference set that is updated by both primal and dual solutions until no more combined (new) solutions can incorporate the reference set.

The RAMP framework relies on the fact that it can incorporate different methods, as components, for building more advanced search strategies. In his paper, Rego [100] presents the general PD-RAMP framework, suggesting Scatter Search (SS) as a possible example for the main component of the primal search, among other approaches. We chose Scatter Search as the evolutionary approach for the basis of our PD-RAMP Primal Method. Our SS is based on the procedure proposed by Laguna e Martí [77]. The Primal-Dual approach uses information provided by the dual method to generate an initial population of solutions to be used by the SS procedure. This SS procedure is divided into the following methods:

Scatter Search Procedure

1. Initial Phase
2. Generate initial population (Reference Set)
3. Apply improvement method
4. Apply Reference Set update method
5. Scatter Search Phase
6. **while** (new solutions are generated in Reference Set)
7. Apply subset generation method
8. Apply solutions combination method
9. Apply improvement method
10. Apply Reference Set update method
11. **end while**

Our PD-RAMP algorithm, starts by populating the pool of solutions with solutions obtained by solving the relaxed problem. The dual solutions are subjected to a Projection Method, that projects them to the primal solutions space, where they are improved using an Improvement Method. Next, the solutions are submitted to a Reference Set Update Method that analyzes (according to the quality or diversification criteria) if they can be included in the Reference Set. Then, the Subset Generation Method creates three sets of solutions (coming from the Reference Set) to be combined by the Combination Method. The objective is to generate new solutions, aiming to explore new regions of the solutions space that traditionally are not explored. An Improvement Method

improves the solution provided by the Combination Method and the improved solution is included in the Solutions Pool.

Figure 5 illustrates the general PD-RAMP model for solving the CSApHLP.

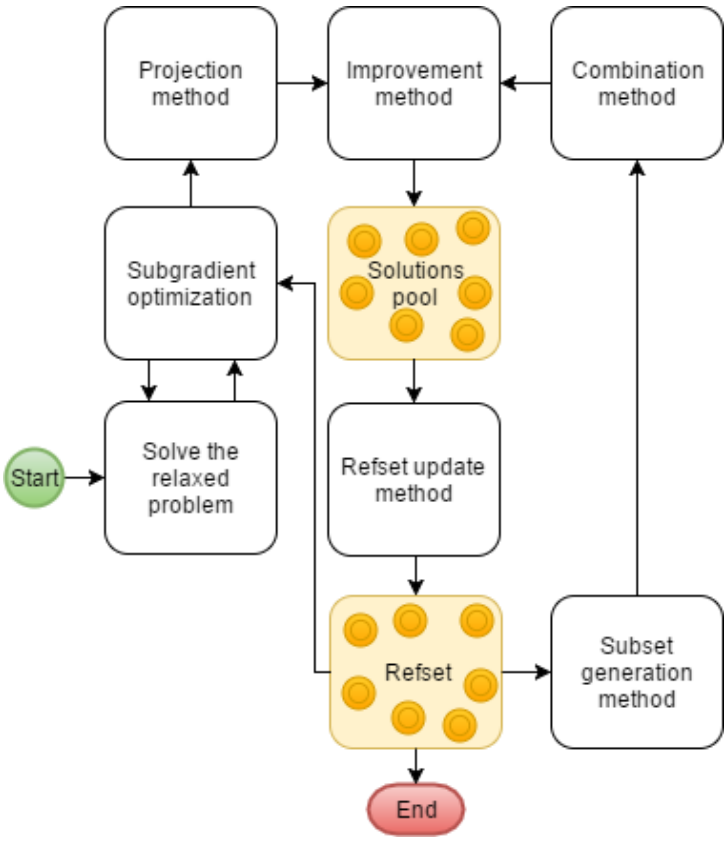

Figure 5: PD-RAMP algorithm overview.

Specifically, each component of the SS works in the following way. The **Solutions Pool** is composed by improved solutions coming from the Projection Method. These diversified solutions are ordered according to quality (increasing order of the objective function value). Every solution has an *id* that identifies the iteration that produced that solution.

The **Reference Set (RefSet)** is composed by S solutions and is divided into three subsets ($S_1, S_2, S_3 \in S$). The **RefeSet Update Method** populates these subsets as follows. The first subset, S_1, has a maximum of two solutions that are considered the best ones (the elite solutions) in terms of the objective function value. The second subset, S_2, contains elite and diversified solutions, up to a maximum of three. The remaining solutions (up to five) are in S_3 and are classified as diversified and the worst in terms of quality.

The **Subset Generation Method** operates on the reference set to produce subsets of solutions that will serve as the basis for the **Combination Method**. The typical implementation of this method is to generate all possible pairs of solutions that contain at least one new solution, although the generation of larger subsets can be considered. A solution is considered new if it has not yet been subjected to the **Combination Method**. In the proposed algorithm, the subset randomly chooses (between zero and the algorithm's iteration) a maximum of five solutions from the Reference Set.

The **Combination Method** combines solutions, two by two, based on a threshold given by the objective function value of each solution. When it finds the same open hubs in both solutions, and one of the solutions have a threshold value higher than a predefined one, then that hub will be opened in the new combined solution. This is based on the idea that choosing open hubs that belong to better solutions (in terms of the objective function value) is more likely to generate elite solutions.

An **Improvement Method** (the same used by Dual-RAMP) tries to improve each solution produced by the Combination Method and includes the resulting solution in the Solutions Pool.

The **Reference Set Update Method** is called again to replace old solutions by new improved solutions and populate the Reference Set as previously explained.

The PD-RAMP algorithm alternates between the dual and primal methods until they fail to improve the best solution found. In addition to the stopping criteria of Dual-RAMP, PD-RAMP stops whenever no new solutions are included in the Reference Set.

8.4. Computational Results

To assess the performance of our RAMP algorithms we run extensive tests on a set of standard AP (Australia Post) instances that result from the mail flows in an Australian city and contain up to 200 nodes. These instances are available at Beasley's OR-Library [11] and were introduced by Ernst and Krishnamoorthy [36]. The flow is asymmetric ($W_{ij} \neq W_{ji}$) and the ratios are defined as $\chi = 3, \alpha = 0.75$ and $\delta = 2$. The dataset has 56 instances grouped by size (small, medium

and large) and with two different types of hubs setup costs and capacities. We have tight (T) setup costs, when the cost increases with the amount of flow generated in the node, and loose (L) setup costs, when the instances do not have this characteristic (the tight setup costs are supposed to be more difficult to solve). As for hubs capacities, we have a similar representation, depending on how tightly (T) or loosely (L) constrained the instances are. Each group consists of four instances that correspond to all four possible combinations of setup costs and capacities (LL, LT, TL or TT). The fixed number of open hubs, p, varies from 2 to 5 and for some instances we have $p > 2$. More detailed information about the instances can be found in Ernst and Krishnamoorthy [36] or Contreras *et al*. [24].

Both algorithms were coded in C programming language and run on an Intel Pentium I7 2.40 GHz (only one processor was used) with 8GB RAM under Ubuntu operating system. Our RAMP algorithms were compared with the state-of-the-art approaches for the CSApHLP: a Reactive GRASP (RG) [122] and a Lagrangean Relaxation approach (LR) [83]. The Dual-RAMP algorithm was compared with all approaches while PD-RAMP was compared directly with Dual-RAMP. All tables show the average percent deviation from the optimal/best-known solutions (Gap) and the associated computational time (CPU) in seconds. We used the optimal/best-known solutions reported by Lu *et al*. [122] (the authors used the GUROBI solver when possible and for the best-known solutions – marked with "[a]" in the tables – they used the values provided by Ernst and Krishnamoorthy [36]).

The first four tables summarize the results for all small AP data instances grouped by type (LL, LT, TL or TT) and size (from 10 to 50 nodes). The "UB" column displays the best upper bound achieved by Dual-RAMP. The "Gap" column was computed as $(UB - Z^*)/UB*100$ ($Z^*$is the optimal/best-known solution) and the "CPU" column shows the computational time (in seconds) needed to achieve the UB. PD-RAMP's "Gap" and "CPU" results are calculated the same way, except when no Z* value is provided, in which case "Gap" was computed considering the "UB" value obtained by Dual-RAMP (in these cases we wrote "-" for Dual-RAMP and only reported the "Gap" value for PD-RAMP). Columns "Nodes" and "p" indicate the instances size and number of fixed hubs (p has a

range from 2 to 5). LT and TT instances with $nodes >= 25$ and $p = 2$ cannot be solve, since the demand is higher than the capacity of the required number of hubs. The RG and the LR algorithms do not provide results for all the small AP data set ("-" means that no solution is available), nevertheless, we report results for both RAMP versions. "Average 1" specifies the average results for the instances tested by all the algorithms and "Average 2" considers all instances results for both RAMP approaches.

Nodes	*p*	**Z***	RG		LR		**Dual-RAMP**			**PD-RAMP**	
			Gap	CPU	Gap	CPU	**UB**	**Gap**	**CPU**	**Gap**	**CPU**
10	2	230008.5	0.00	0.2	0.00	25.28	**230008.5**	**0.00**	**0.01**	**0.00**	**0.01**
	3	224250.1	0.00	0.1	0.00	2.32	**224250.1**	**0.00**	**0.01**	**0.00**	**0.01**
	4	229172.6	0.00	0.1	0.00	2.31	**229172.6**	**0.00**	**0.01**	**0.00**	**0.01**
	5	239292.3	0.00	0.1	0.00	1.28	**239292.3**	**0.00**	**0.01**	**0.00**	**0.01**
20	2	234691.0	0.00	0.4	0.00	79.23	**234691.0**	**0.00**	**0.09**	**0.00**	**0.08**
	3	239444.2	0.00	0.2	0.00	35.54	**239444.3**	**0.00**	**0.16**	**0.00**	**0.15**
	4	251939.7	0.00	0.3	0.00	38.8	**251939.7**	**0.00**	**0.29**	**0.00**	**0.28**
	5	266745.2	0.00	0.3	0.00	15.18	**266745.2**	**0.00**	**0.14**	**0.00**	**0.17**
25	2	238978.0	0.00	0.6	0.00	190.4	**238978.0**	**0.00**	**0.37**	**0.00**	**0.47**
	3	242437.2	0.00	0.4	0.20	132.3	**242437.2**	**0.00**	**1.20**	**0.00**	**1.31**
	4	252716.6	0.00	0.5	0.00	64.91	**252716.6**	**0.00**	**1.16**	**0.00**	**1.33**
	5	263518.3	0.00	0.6	0.00	74.68	**263518.3**	**0.00**	**0.81**	**0.00**	**1.00**
40	2	241955.71[a]	0.00	0.6	0.00	18.97	**241955.7**	**0.00**	**7.51**	**0.00**	**1.29**
	3	-	-	-	-	-	**244329.4**	**-**	**6.26**	**0.00**	**1.39**
	4	-	-	-	-	-	**255981.6**	**-**	**8.33**	**0.00**	**1.44**
	5	-	-	-	-	-	**269895.5**	**-**	**6.32**	**0.00**	**6.45**
50	2	238520.59	0.00	0.8	0.00	564.4	**238520.6**	**0.00**	**16.90**	**0.00**	**19.70**
	3	-	-	-	-	-	**242768.1**	**-**	**18.20**	**0.00**	**21.35**
	4	-	-	-	-	-	**252536.7**	**-**	**18.50**	**0.00**	**22.16**
	5	-	-	-	-	-	**263740.1**	**-**	**17.30**	**0.00**	**23.74**
		Average 1	**0.00**	**0.35**	**0.02**	**88.97**	**-**	**0.00**	**2.05**	**0.00**	**1.84**
		Average 2	**-**	**-**	**-**	**-**	**-**	**-**	**5.18**	**0.00**	**5.12**

Table 18: LL small instances results.

Nodes	p	Z*	RG		LR		**Dual-RAMP**			**PD-RAMP**	
			Gap	CPU	Gap	CPU	**UB**	**Gap**	**CPU**	**Gap**	**CPU**
10	2	256048.6	0.00	0.00	0.00	46.87	**256048.6**	**0.00**	**0.01**	**0.00**	**0.01**
	3	252973.6	0.00	0.00	0.00	137.65	**252973.6**	**0.00**	**0.02**	**0.00**	**0.02**
	4	250992.3	0.00	0.00	0.00	22.91	**250992.3**	**0.00**	**0.01**	**0.00**	**0.01**
	5	261451.2	0.00	0.03	0.00	38.59	**261451.2**	**0.00**	**0.02**	**0.00**	**0.02**

Nodes	p	Z*	RG		LR		Dual-RAMP			PD-RAMP	
			Gap	CPU	Gap	CPU	UB	Gap	CPU	Gap	CPU
20	2	253517.4	0.00	0.05	2.99	316.97	**253517.4**	**0.00**	**0.42**	**0.00**	**0.55**
	3	257247.7	0.00	0.24	0.14	18.39	**257247.7**	**0.00**	**0.57**	**0.00**	**0.62**
	4	260678.9	0.00	0.20	0.00	280.05	**260678.9**	**0.00**	**0.29**	**0.00**	**0.36**
	5	274975.4	0.00	0.21	0.00	217.75	**274975.4**	**0.00**	**0.46**	**0.00**	**0.59**
25	3	276372.5	0.00	0.44	0.00	1494.3	**276372.5**	**0.00**	**1.02**	**0.00**	**1.26**
	4	278235.0	0.00	0.42	0.00	1862.9	**278235.0**	**0.00**	**1.18**	**0.00**	**1.08**
	5	284952.6	0.00	0.43	0.00	377.67	**284952.6**	**0.00**	**1.28**	**0.00**	**1.36**
40	3	272218.32[a]	0.00	1.19	5.36	289.24	**272218.3**	**0.00**	**7.25**	**0.00**	**8.38**
	4	-	-	-	-	-	**283182.2**	**-**	**7.72**	**0.00**	**9.02**
	5	-	-	-	-	-	**297621.3**	**-**	**7.82**	**0.00**	**9.97**
50	3	-	-	-	-	-	**276725.6**	**-**	**17.36**	**0.00**	**19.27**
	4	272897.49	0.91	2.01	6.3	49.98	**272897.4**	**0.00**	**17.87**	**0.00**	**21.65**
	5	-	-	-	-	-	**283003.2**	**-**	**19.05**	**0.00**	**23.06**
Average 1			**0.07**	**0.40**	**1.14**	**396.41**	**-**	**0.00**	**2.34**	**0.00**	**2.76**
Average 2			**-**	**-**	**-**	**-**	**-**	**-**	**4.84**	**0.00**	**5.72**

Table 19: LT small instances results.

Nodes	*p*	**Z***	RG		LR		Dual-RAMP			PD-RAMP	
			Gap	CPU	Gap	CPU	UB	Gap	CPU	Gap	CPU
10	2	264544.0	0.00	0.2	0.00	12.59	**264544.0**	**0.00**	**0.01**	**0.00**	**0.01**
	3	263399.9	0.00	0.2	0.00	2.44	**263399.9**	**0.00**	**0.01**	**0.00**	**0.01**
	4	269074.3	0.00	0	0.00	0.72	**269074.3**	**0.00**	**0.01**	**0.00**	**0.01**
	5	281327.3	0.00	0.1	0.00	0.3	**281327.3**	**0.00**	**0.01**	**0.00**	**0.01**
20	2	271128.2	0.00	0.7	0.00	54.09	**271128.2**	**0.00**	**0.15**	**0.00**	**0.13**
	3	281304.8	0.00	0.3	0.4	29.51	**281304.8**	**0.00**	**0.17**	**0.00**	**0.23**
	4	295223.9	0.00	0.3	0.00	22.98	**295223.9**	**0.00**	**0.40**	**0.00**	**0.42**
	5	310122.8	0.00	0.4	0.00	3.5	**310122.8**	**0.00**	**0.17**	**0.00**	**0.30**
25	2	310317.6	0.00	0.5	0.00	113.4	**310317.6**	**0.00**	**0.38**	**0.00**	**0.41**
	3	328092.6	0.00	0.3	0.00	90.56	**328092.6**	**0.00**	**1.25**	**0.00**	**1.26**
	4	347416.8	0.00	0.5	0.00	74.59	**347416.8**	**0.00**	**1.35**	**0.00**	**1.34**
	5	368288.6	0.00	0.5	0.1	31.63	**368288.6**	**0.00**	**0.45**	**0.00**	**0.63**
40	2	298919.01[a]	0.00	0.6	0.00	231.1	**298919.0**	**0.00**	**7.20**	**0.00**	**8.60**
	3	-	-	-	-	-	**316445.4**	**-**	**3.96**	**0.00**	**7.78**
	4	-	-	-	-	-	**335857.8**	**-**	**4.92**	**0.00**	**6.61**
	5	-	-	-	-	-	**361113.1**	**-**	**5.81**	**0.00**	**9.05**
50	2	319015.77	0.00	1.3	0.00	381.8	**319015.7**	**0.00**	**17.40**	**0.00**	**19.55**
	3	-	-	-	-	-	**330844.9**	**-**	**13.40**	**0.00**	**19.66**
	4	-	-	-	-	-	**351213.5**	**-**	**9.62**	**0.00**	**18.26**
	5	-	-	-	-	-	**381587.1**	**-**	**17.80**	**0.00**	**21.50**
Average 1			**0.00**	**0.42**	**0.04**	**74.95**	**-**	**0.00**	**2.07**	**0.00**	**2.35**

Nodes	*p*	**Z***	RG		LR		**Dual-RAMP**			**PD-RAMP**	
			Gap	CPU	Gap	CPU	**UB**	**Gap**	**CPU**	**Gap**	**CPU**
		Average 2	**-**	**-**	**-**	**-**	**-**	**-**	**4.22**	**0.00**	**5.79**

Table 20: TL small instances results.

Nodes	p	Z*	RG		LR		**Dual-RAMP**			**PD-RAMP**	
			Gap	CPU	Gap	CPU	**UB**	**Gap**	**CPU**	**Gap**	**CPU**
10	2	264544.0	0.00	0.01	0.00	2.22	**264544.0**	**0.00**	**0.01**	**0.00**	**0.01**
	3	2633 99.9	0.00	0.03	0.00	2.25	**263399.9**	**0.00**	**0.01**	**0.00**	**0.01**
	4	269074.3	0.00	0.00	0.00	0.48	**269074.3**	**0.00**	**0.01**	**0.00**	**0.01**
	5	281327.3	0.00	0.07	0.00	0.28	**281327.3**	**0.00**	**0.01**	**0.00**	**0.01**
20	2	329068.6	0.00	0.01	9.8	54.97	**329068.6**	**0.00**	**0.53**	**0.00**	**0.56**
	3	296035.4	0.00	0.25	0.00	70.67	**296035.4**	**0.00**	**0.17**	**0.00**	**0.21**
	4	306296.3	0.00	0.28	0.00	9.25	**306296.3**	**0.00**	**0.12**	**0.00**	**0.10**
	5	325568.5	0.00	0.32	0.58	10.97	**325568.5**	**0.00**	**0.11**	**0.00**	**0.11**
25	3	348369.1	0.00	0.33	0.00	99.59	**348369.1**	**0.00**	**1.18**	**0.00**	**1.29**
	4	369576.8	0.00	0.47	0.00	67.05	**369576.8**	**0.00**	**1.21**	**0.00**	**1.39**
	5	391996.5	0.00	0.47	0.00	38.47	**391996.5**	**0.00**	**1.25**	**0.00**	**1.44**
40	3	354874.10[a]	1.15	1.16	4.66	190.96	**357865.3**	**0.80**	**7.25**	**0.00**	**8.02**
	4	-	-	-	-	-	**366720.9**	**-**	**7.34**	**0.00**	**8.85**
	5	-	-	-	-	-	**384295.3**	**-**	**7.48**	**0.00**	**9.00**
50	3	-	-	-	-	-	**425302.3**	**-**	**17.18**	**-1.22**	**18.74**
	4	417440.99	0.86	2.45	10.1	28.55	**419293.4**	**0.40**	**18.30**	**0.08**	**19.82**
	5	-	-	-	-	-	**440790.1**	**-**	**18.47**	**-2.22**	**21.61**
		Average 1	**0.15**	**0.45**	**1.93**	**44.29**	**-**	**0.10**	**2.32**	**0.01**	**2.54**
		Average 2	**-**	**-**	**-**	**-**	**-**	**-**	**4.74**	**-0.20**	**5.36**

Table 21: TT small instances results.

Both RAMP approaches managed to achieve excellent quality results for all small instances, clearly outperforming the other two algorithms. PD-RAMP produced a 0.01% average deviation from the optimal/best-Known solutions in reduced computational times. Dual-RAMP achieved all optimal solutions, except for instances 40TT ($p = 3$) and 50TT ($p = 4$), providing a "Gap" of 0.84% and 0.44%, respectively. PD-RAMP was able to achieve better results than Dual-RAMP, only failing optimality for one instance, the 50TT ($p = 4$), and significantly improving the corresponding Dual-RAMP result.

The next four tables (Table 22 to Table 25) show the results for the medium instances (from 60 to 100 nodes) for which no solution is available in the

literature. As far as we know, these are the first results reported for these CSApHLP instances. The LT and TT instances with $p = 2$ cannot be solved, due to insufficient hubs capacity to fulfill all nodes demand. In these tables, we can see a direct comparison between Dual-RAMP and PD-RAMP. The "UB" column displays the best upper bound achieved by Dual-RAMP and the "Gap" column was computed as (UB - Z^*)/ UB*100, where Z^* corresponds to the "UB" value provided by Dual-RAMP.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
60	2	225917.16	34.12	0.00	39.25
	3	230862.05	35.53	0.00	42.97
	4	237439.83	35.84	0.00	44.19
	5	245404.58	36.09	0.00	44.55
70	2	236900.33	61.95	-0.04	70.59
	3	238666.31	63.16	0.00	77.94
	4	245426.59	64.93	-0.72	82.91
	5	255221.64	65.07	0.00	87.19
75	2	239130.75	81.50	-0.46	93.36
	3	245441.67	79.90	-0.37	99.62
	4	253886.67	84.33	0.00	110.77
	5	269010.34	87.12	-1.5	120,00
90	2	224195.69	90.04	0.00	196.07
	3	228453.02	87.18	0.00	212.60
	4	240898.81	83.50	-1.18	226.67
	5	250477.88	91.20	0.00	230.22
100	2	250508.59	83.50	0.00	103.30
	3	246713.95	90.20	0.00	114,11
	4	255507.17	86.50	0.00	125,22
	5	271650.28	97.17	-1.33	133,87
		Average	**90.62**	**-0.29**	**112.77**

Table 22: LL medium instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
60	3	269921.0	34.74	0.33	40.14
	4	257854.6	36.12	-1.46	44.52

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
	5	263252.9	38.07	-0.15	46.94
70	3	257763.6	61.5	0.00	68.86
	4	258028.3	63.25	-0.39	79.17
	5	263814.9	65.13	-0.18	83.41
75	3	256188.1	81.31	0.00	91.30
	4	259940.9	85.3	0.00	99.60
	5	273778.8	88.42	0.27	108.66
90	3	249050.7	160.8	0.12	203.17
	4	249609.9	166.4	0.16	220.70
	5	261319.4	169	-0.24	221.76
100	3	269819.0	96.69	-3.86	102.97
	4	270439.6	96.26	-1.81	111.98
	5	278304.3	100.5	-0.57	125.44
		Average	**89.56**	**-0.52**	**109.91**

Table 23: LT medium instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
60	2	252496.7	35.03	0.00	39.75
	3	263732.0	35.16	0.00	40.09
	4	281642.9	34.6	0.00	41.45
	5	302844.4	34.9	0.00	42.04
70	2	271283.8	61.11	0.00	71.26
	3	283507.3	61.51	0.00	76.01
	4	303373.2	63.08	0.00	79.88
	5	326569.3	65.02	0.00	81.30
75	2	303363.50	73.52	0.00	91.98
	3	306864.30	82.47	-0.42	93.29
	4	325114.40	84.67	-0.14	100.63
	5	344063.10	83.72	0.00	106.04
90	2	298956.20	86.50	0.01	188.31
	3	281919.20	84.36	-0.03	203.98
	4	286717.60	82.50	0.00	208.97
	5	309330.50	88.40	0.00	213.46

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
100	2	385234.80	91.10	-5.90	104.22
	3	365648.66	86.20	0.60	113.20
	4	389885.91	81.20	-2.05	129.76
	5	410317.28	90.52	-0.04	133.94
		Average	**70.28**	**-0.40**	**107.98**

Table 24: TL medium instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
60	3	351669	34.43	-0.03	39.15
	4	359255.7	34.69	0.06	41.81
	5	373591.4	34.99	0.01	42.40
70	3	457218.5	62.08	0.08	67.11
	4	387397.9	64.77	0.00	74.84
	5	408468.3	65.54	-0.39	78.76
75	3	372936.1	81.24	0.00	89.32
	4	347371.5	84.88	-0.05	94.15
	5	363528.7	86.75	-0.02	102.31
90	3	396614.3	162.8	-0.16	184.19
	4	349092.1	166.4	-3.20	208.12
	5	357883.9	170.2	-1.70	217.81
100	3	576133.3	94.93	-7.05	103.54
	4	491686.4	99.42	-2.40	111.53
	5	488904.6	104.1	3.22	123.05
		Average	**76.37**	**-0.78**	**105.21**

Table 25: TT medium instances results.

The PD-RAMP algorithm managed to find better solutions than Dual-RAMP with a slightly higher computational effort (as expected) due to the intensification of the primal side exploration. Nevertheless, the increase in solutions quality clearly compensates the higher computational times.

Table 26 to Table 29 summarize the results for the largest instances (from 125 to 200 nodes) for which, as far as we know, there are no solutions available in the literature.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
125	2	243092.5	259.93	-0.27	270.16
	3	240667.7	255.01	-0.32	290.52
	4	250668.3	258.07	-2.19	313.79
	5	264679.1	262.53	-3.47	332.16
150	2	234765.4	469.05	0.00	486.16
	3	238348.8	477.72	-0.29	550.29
	4	244912.6	483.46	-0.19	596.85
	5	258623.2	493.84	-1.33	650.66
175	2	230477.10	885.90	-0.44	1017.01
	3	229743.70	899.59	-1.07	912.97
	4	239710.90	906.06	-0.74	1043.93
	5	252320.40	916.23	0.00	1096.70
200	2	249971.02	1149.26	-4.34	1220.94
	3	231069.52	1170.69	0.00	1302.51
	4	236508.92	1185.55	-0.61	1341.37
	5	249619.03	1191.76	-0.33	1454.80
		Average	**704.04**	**-0.97**	**805.05**

Table 26: LL large instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
125	3	270569.7	251.76	0.13	271,09
	4	261564.5	257.92	-4.09	295,61
	5	257604.6	265.17	-0.02	302,73
150	3	293488.5	469.52	-4.04	500,27
	4	252894.3	478.72	-0.40	521,29
	5	260356.8	494.02	-1.32	561,96
175	3	261998.7	880.29	0.00	875,85
	4	256940.3	906.39	-1.99	956,27
	5	263236.8	919.34	-0.93	1049,41
200	3	292306.9	1163.4	-5.35	1185,94
	4	277819.9	1173.5	-2.95	1323,78
	5	283591.1	1209.7	-0.89	1393,36
		Average	**705.81**	**-1.82**	**769.80**

Table 27: LT large instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
125	2	264860.9	252.08	-7.45	263.52
	3	265706	254.11	-1.81	280.43
	4	280483.8	258.84	0.00	295.59
	5	313370.8	262.36	-3.05	306.81
150	2	279501.9	469.06	-3.53	499.52
	3	263356.2	476.72	0.00	532.79
	4	277607.2	480.93	-1.37	569.20
	5	297615.9	487.43	-2.90	602.54
175	2	246381.63	881.91	0.00	905.77
	3	255948.78	889.43	-0.66	968.92
	4	270384.16	900.02	-1.82	1062.50
	5	288107.16	917.62	-2.25	1129.04
200	2	290393.53	1155.55	-2.69	1202.00
	3	288963.38	1175.50	-5.68	1251.73
	4	296948.19	1137.30	-3.03	1380.06
	5	298996.53	1173.99	-0.48	1455.20
		Average	**698.30**	**-2.29**	**794.10**

Table 28: TL large instances results.

Nodes	p	Dual-RAMP		PD-RAMP	
		UB	CPU	Gap	CPU
125	3	353995.8	251.19	-11.06	264.07
	4	326708.6	258.51	-11.17	288.93
	5	329872.1	261.5	-5.11	306.32
150	3	339679.9	473.88	-0.02	499.50
	4	328340.9	484.97	1.38	530.81
	5	378218.8	492.4	-10.43	580.13
175	3	339850.4	895.99	-6.87	933.67
	4	337573	889.42	-5.63	985.66
	5	335327.2	909.64	-0.93	1062.69
200	3	514873.6	1158.2	-0.54	1162.09
	4	334586.9	1179.4	-13.16	1325.22
	5	328327.2	1195.4	-10.44	1410.41
		Average	**704.21**	**-6.16**	**779.12**

Table 29: TT large instances results.

For the largest instances in the literature, PD-RAMP produced better results than Dual-RAMP in almost every instance (and for all combinations of size and type). The intensive exploration of the primal side with the use of Scatter Search, led to an average increase in the computational effort of 10%, which is perfectly acceptable considering the gains in quality.

In conclusion, the significant improvement in solutions quality with reduced computational times renders PD-RAMP the best RAMP approach.

Table 30 to Table 35 show the results for the CSApHLP in which no fixed costs are considered (in the literature, this problem is usually refered as CSApHMP – "M" stands for "Median"). We decided to test our PD-RAMP algorithm on this problem, but it is important to notice that the CSApHMP is a different problem, since the sum $\sum_{k\in N} f_k Z_{kk}$ is removed from the objective function formulation. The aim of the present study was addressing the CSApHLP, nevertheless, we wanted to see how PD-RAMP would perform against other approaches specially designed for the CSApHMP.

We only compare PD-RAMP with a Genetic Algorithm proposed by Stanimirović (GA) [116] due to the fact that the hybrid GRASP-Path Relinking algorithm proposed by Perez *et al*. [97] assumes different parameters for the discount factor, that are not provided by the author, and thus it cannot be compared in terms of the objective function value. We used the same instances previously described for the CSApHLP with the slight difference that it is not necessary to address TL and TT instances, since they are the same as LL and LT (when the fixed costs associated with the hubs are ignored), respectively. In addition, new fixed number of hubs ($p \in \{10,15,20\}$) were used for some instance sizes, to allow a direct comparison with the GA algorithm. GA reports an average "Gap" value obtained after 20 runs on each instance. Our algorithm obtained the reported solutions with a single run.

Table 30 and Table 31 show the results for the small instances (from 10 to 50 nodes).

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap**	**CPU**
10	2	0.00	0.37	0.00	0.01
	3	0.00	0.50	0.00	0.06

Nodes	p	GA		PD-RAMP	
		Gap	CPU	Gap	CPU
	4	0.00	0.52	0.00	0.02
	5	0.00	0.64	0.00	0.01
20	2	-	-	0.00	0.11
	3	0.00	0.99	0.00	0.20
	4	0.00	1.08	0.00	0.42
	5	0.00	1.49	0.00	0.66
25	2	0.01	1.93	0.00	0.26
	3	0.00	0.84	0.00	0.81
	4	0.00	1.32	0.00	0.43
	5	0.14	1.56	0.00	1.43
	10	0.01	1.98	0.00	1.37
40	2	0.00	2.64	0.00	2.03
	3	0.00	1.50	0.00	8.32
	4	0.00	2.48	0.12	9.05
	5	0.00	3.22	0.00	9.35
	10	0.01	3.74	0.00	10.09
50	2	0.05	5.40	0.00	9.56
	3	0.00	2.28	0.00	19.24
	4	0.00	3.64	0.00	21.43
	5	0.00	4.22	0.00	23.64
	10	0.05	5.11	0.00	23.37
	Average	**0.02**	**2.35**	**0.005**	**7.23**

Table 30: LL small instances results for the CSApHMP.

Nodes	p	GA		PD-RAMP	
		Gap	CPU	Gap	CPU
10	2	0.00	0.36	0.00	0.01
	3	0.00	0.49	0.00	0.02
	4	0.00	0.51	0.00	0.01
	5	0.00	0.67	0.00	0.02
20	2	0.00	0.00	0.00	0.59
	3	0.50	0.98	0.00	0.63
	4	0.00	1.03	0.00	0.17
	5	0.00	1.26	0.00	0.12
25	2	0.00	1.70	0.00	0.20

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap**	**CPU**
	3	0.52	1.14	0.00	1.26
	4	0.22	1.39	0.17	1.35
	5	0.39	1.99	0.00	0.22
	10	0.00	2.24	0.00	1.43
40	2	5.38	1.95	0.00	7.98
	3	0.00	2.85	0.00	8.74
	4	0.11	3.29	0.00	9.38
	5	0.07	5.45	0.00	10.12
	10	1.13	3.11	0.00	18.89
50	2	0.65	5.13	0.00	20.75
	3	0.08	5.17	0.00	22.84
	4	0.35	8.17	0.00	29.42
	5	0.00	0.36	0.00	0.01
	10	0.00	0.49	0.00	0.02
	Average	**0.45**	**2.32**	**0.008**	**6.38**

Table 31: LT small instances results for the CSA$\boldsymbol{p}$HMP.

For the small instances, PD-RAMP achieved almost every optimal solution with a single run, obtaining average "Gap" values of 0.005% and 0.008% for the LL and LT instances, respectively. PD-RAMP only failed to find the optimal solution for instances 25LT ($p = 4$) and 40LL ($p = 4$) obtaining a deviation from the optimal solution of 0.12% and 0.17%, respectively. In terms of computational time, no direct comparison can be made due to the use of different machines, but nonetheless, the PD-RAMP algorithm takes slightly longer to return the solutions. If we consider the best upper bound found within the 20 runs for each instance, GA achieved every optimal solution, except for 20LT ($p = 2$), for which GA could not produce any solutions. Nevertheless, PD-RAMP definitely outperformed the GA approach, considering that it only took a single run to produce these high-quality results.

The next two tables show the results for the medium instances (from 60 to 100 nodes). GA only provides results for the 100 nodes instances and for the LT instances it does not report results for $p = 3$. We present results for the PD-

RAMP algorithm on all instances and, as far as we know, the solutions we provide for instances with up to 90 nodes are the first in the literature for the CSApHMP. The "Gap" column was computed as (UB – B*)/ UB $*$100 (B* is the best solution provided by Stanimirović [116] within 20 runs of the GA algorithm) and the "Best UB" column displays the best upper bound achieved by the PD-RAMP algorithm.

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap/Best UB**	**CPU**
60	2	-	-	180651.08	38.57
	3	-	-	163305.23	42.41
	4	-	-	146874.91	47.94
	5	-	-	133203.48	51.15
70	2	-	-	185676.61	69.20
	3	-	-	165027.38	78.07
	4	-	-	146717.19	87.96
	5	-	-	136903.77	92.88
75	2	-	-	186520.73	96.62
	3	-	-	165129.27	105.83
	4	-	-	151754.89	122.20
	5	-	-	139666.64	133.14
	10	-	-	179889.58	186.81
90	2	-	-	160735.81	208.59
	3	-	-	145427.11	237.21
	4	-	-	136089.58	253.30
	5	-	-	180651.08	238.57
100	2	0.04	9.24	0.00	100.11
	3	0.01	16.60	0.00	112.65
	4	0.01	16.57	0.00	125.46
	5	0.09	18.87	0.04	140.23
	10	0.35	29.53	0.56	234.45
	15	0.57	36.09	2.40	488.25
	20	0.61	43.29	2.03	598.53
	Average	**0.24**	**24.31**	**0.72**	**257.10**

Table 32: LL medium instances results for the CSA$\boldsymbol{p}$HMP.

Nodes	p	GA		PD-RAMP	
		Gap	CPU	Gap/Best UB	CPU
60	3	-	-	182418.19	39.00
	4	-	-	152147.27	42.81
	5	-	-	138173.05	46.36
70	3	-	-	177547.09	23.48
	4	-	-	152934.38	66.85
	5	-	-	140011.56	83.13
75	3	-	-	174932.08	92.81
	4	-	-	150761.66	98.61
	5	-	-	139786.94	106.01
90	3	-	-	181171.91	185.77
	4	-	-	152525.38	203.40
	5	-	-	139943.44	216.20
100	3	-	-	180294.08	99.43
	4	0.49	15.50	1.22	109.35
	5	0.72	21.67	1.32	116.05
	10	1.19	32.12	1.02	411.25
	15	0.60	34.64	1.43	421.25
	20	0.69	42.90	3.61	600.15
Average		**0.74**	**29.37**	**1.72**	**331.61**

Table 33: LT medium instances results for the CSA$\boldsymbol{p}$HMP.

For the medium instances, PD-RAMP did not achieve the same quality results as for the small instances. Specifically, for the 100 nodes LL instances, the proposed algorithm obtained an average "Gap" value of 0.72% in 257.10 seconds, and for the 100 nodes LT instances, PD-RAMP achieved an average "Gap" value of 1.72% in 331.61 seconds. For these instances, the GA approach outperformed PD-RAMP.

The last two tables show the results for the large instances (from 125 to 200 nodes). GA only provides results for the 200 nodes instances and for the LT instances it does not report results for $p = 3$. We present results for the PD-RAMP algorithm on all instances and, as far as we know, the solutions we provide for instances with up to 175 nodes are the first in the literature for the CSApHMP.

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap**	**CPU**
125	2	-	-	180941.19	261.01
	3	-	-	161117.13	287.60
	4	-	-	146173.13	316.56
	5	-	-	137372.63	348.36
150	2	-	-	180898.78	479.64
	3	-	-	161490.34	527.37
	4	-	-	146521.23	581.72
	5	-	-	138648.61	635.23
175	2	-	-	182292.72	899.39
	3	-	-	162584.63	985.85
	4	-	-	147361.13	1067.90
	5	-	-	139998.38	1163.57
200	2	0.02	58.92	2.59	1173.88
	3	0.00	93.14	1.66	1258.73
	4	0.02	107.52	0.01	1359.09
	5	0.26	118.70	**-0.06**	**1454.15**
	10	0.86	183.00	1.55	1521.13
	15	0.61	223.73	3.05	1912.52
	20	0.55	261.82	1.93	1984.52
Average		**0.33**	**149.55**	**1.53**	**1523.43**

Table 34: LL large instances results for the CSApHMP.

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap**	**CPU**
125	3	-	-	177984.23	253.61
	4	-	-	157683.70	277.96
	5	-	-	144252.77	299.48
150	3	-	-	214943.67	465.35
	4	-	-	157954.09	500.39
	5	-	-	142694.05	547.63
175	3	-	-	188608.69	876.53
	4	-	-	153247.48	961.22
	5	-	-	142119.25	1032.05
200	3	-	-	182663.67	1169.53
	4	0.37	115.71	**-0.47**	**1242.40**

Nodes	p	GA		PD-RAMP	
		Gap	**CPU**	**Gap**	**CPU**
	5	1.70	139.99	**-0.03**	**1407.18**
	10	2.51	204.92	0.74	1545.54
	15	1.38	223.06	2.54	1635.45
	20	1.04	260.03	4.96	1895.22
Average		**1.40**	**188.74**	**1.55**	**1545.16**

Table 35: LT large instances results for the CSA$\boldsymbol{p}$HMP.

PD-RAMP required a significantly greater computational effort to solve the largest instances in the literature for this version of the Hub Location Problem. Nevertheless, the PD-RAMP algorithm managed to achieve new best-known solutions for instances 200LL ($p = 5$), 200LT ($p = 4$) and 200LT ($p = 5$) with an upper bound of 140472.73, 157422.16 and 143377.67, respectively.

8.5. Conclusions

In this work, we present two RAMP approaches for solving the CSApHLP. A Dual-RAMP algorithm (that combines Lagrangean Relaxation with subgradient optimization and local search) and a PD-RAMP algorithm (that integrates the Dual-RAMP algorithm with a Scatter Search procedure to intensify the search on the primal side of the problem).

Both algorithms produced excellent results on a standard testbed of small, medium and large instances, outperforming the current state-of-the-art algorithms for this problem. Since no solutions could be found in the literature for the medium and large instances, we introduced the first solutions for all those instances with sizes ranging from 60 to 200 nodes.

Comparing both RAMP approaches, it is clear that PD-RAMP shows equal or better results than Dual-RAMP for almost every instance. The significant gains in solutions quality obtained in reduced computational times define PD-RAMP as the best approach.

The combination of mathematical relaxation techniques with adaptive memory techniques and their interconnection with suitable research algorithms proved to be a very effective and efficient approach for solving problems of this type.

References

[1] B. M. Akinc, Umit and Khumawala, “An Efficient Branch and Bound Algorithm for the Capacitated Warehouse Location Problem,” *Manage. Sci.*, vol. 23, no. 6, pp. 585–594, 1977.

[2] W. G. De Almeida, H. H. Yanasse, and E. L. F. Senne, “Uma abordagem exata para problema de localização de concentradores capacitado,” in *43 Simpósio Brasileiro de Pesquisa Operacional*, 2011, pp. 2192–2203.

[3] W. Almeida, E. L. F. Senne, and H. Yanasse, “Abordagens meta-heurísticas para o problema de localização de concentradores com restrições de capacidade,” in *X Worcap*, 2010, p. 12.

[4] S. Alumur and B. Y. Kara, “Network hub location problems: The state of the art,” *Eur. J. Oper. Res.*, vol. 190, no. 1, pp. 1–21, 2008.

[5] P. Avella and M. Boccia, “A cutting plane algorithm for the capacitated facility location problem,” *Comput. Optim. Appl.*, vol. 43, no. 1, pp. 39–65, Dec. 2009.

[6] P. Avella, M. Boccia, A. Sforza, I. Vasil’ev, and I. Vasil’ev, “An effective heuristic for large-scale capacitated facility location problems,” *J. Heuristics*, vol. 15, no. 6, pp. 597–615, May 2008.

[7] J. Baker, “Adaptive selection methods for genetic algorithms,” *Proc. 1st Int. Conf. Genet. Algorithms*, pp. 101–111, 1985.

[8] J. E. Beasley, “Obtaining test problems via internet,” *J. Glob. Optim.*, vol. 8, no. 4, pp. 429–433, 1996.

[9] J. E. Beasley, “An algorithm for solving large capacitated warehouse location problems,” *Eur. J. Oper. Res.*, vol. 33, no. 3, pp. 314–325, 1988.

[10] J. E. Beasley, “Lagrangean heuristics for location problems,” *Eur. J. Oper. Res.*, vol. 65, no. 3, pp. 383–399, 1993.

[11] J. Beasley, “OR-Library: distributing test problems by electronic mail,” *J. Oper. Res. Soc.*, vol. 65, pp. 1069–1072, 1990.

[12] J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numer. Math.*, vol. 4, no. 1, pp. 238–252, Dec. 1962.

[13] O. Bilde and J. Krarup, “Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem,” in *Annals of Discrete Mathematics*, 1977, pp. 79–97.

[14] C. T. Bornstein, “An ADD/DROP procedure for the capacitated plant location problem,” *Pesqui. Operacional*, pp. 151–162, 2003.

[15] C. T. C. Bornstein and H. H. B. Azlan, “The use of reduction tests and simulated annealing for the capacitated plant location problem,” *Locat. Sci.*, vol. 6, no. 1998, pp. 67–81, 1998.

[16] D. L. Bryan and M. E. O’Kelly, “Hub-and-Spoke Networks in Air Transportation: An Analytical Review,” *J. Reg. Sci.*, vol. 39, no. 2, pp. 275–295, 1999.

[17] G. Cabrera G., E. Cabrera, R. Soto, L. J. M. Rubio, B. Crawford, and F. Paredes, “A Hybrid Approach Using an Artificial Bee Algorithm with Mixed Integer Programming Applied to a Large-Scale Capacitated Facility Location Problem,” *Math. Probl. Eng.*, vol. 2012, p. 14, 2012.

[18] J. F. Campbell, “Hub Location and the p-Hub Median Problem,” *Oper. Res.*, vol. 44, no. 6, pp. 923–935, 1996.

[19] J. F. Campbell, “A survey of network hub location,” *Stud. Locat. Anal.*, vol. 6, pp. 31–49, 1994.

[20] J. F. Campbell, “Integer programming formulations of discrete hub location problems,” *Eur. J. Oper. Res.*, vol. 72, no. 2, pp. 387–405, 1994.

[21] J. F. Chen, “A heuristic for the capacitated single allocation hub location problem,” in *Lecture Notes in Electrical Engineering*, vol. 5, 2008, pp. 185–196.

[22] F. a. Chudak and D. P. Williamson, “Improved approximation algorithms for capacitated facility location problems,” *Math. Program.*, vol. 102, no. 2, pp. 207–222, May 2004.

[23] I. Contreras, J. A. Díaz, and E. Fernández, “Lagrangean relaxation for the capacitated hub location problem with single assignment,” *OR Spectr.*, vol. 31, no. 3, pp. 483–505, 2009.

[24] I. Contreras, J. A. Díaz, and E. Fernández, “Lagrangean relaxation for the capacitated hub location problem with single assignment,” *OR Spectr.*, vol. 31, no. 3, pp. 483–505, Jun. 2009.

[25] G. Cornuéjols, R. Sridharan, and J. Thizy, “A comparison of heuristics and relaxations for the capacitated plant location problem,” *Eur. J. Oper. Res.*, vol. 50, pp. 280–297, 1991.

[26] I. Correia, S. Nickel, and F. Saldanha-da-Gama, “Multi-product Capacitated Single-Allocation Hub Location Problems: Formulations and Inequalities,” *Networks Spat. Econ.*, vol. 14, no. 1, pp. 1–25, 2014.

[27] I. Correia, S. Nickel, and F. Saldanha-da-Gama, “The capacitated single-allocation hub location problem revisited: A note on a classical formulation,” *Eur. J. Oper. Res.*, vol. 207, no. 1, pp. 92–96, 2010.

[28] M. da G. Costa, M. E. Captivo, and J. Clímaco, “Capacitated single allocation hub location problem-A bi-criteria approach,” *Comput. Oper. Res.*, vol. 35, no. 11, pp. 3671–3695, 2008.

[29] M. S. Daskin, L. V. Snyder, and R. T. Berger, “Facility Location in Supply Chain Design,” in *Logistics Systems: Design and Optimization*, no. 3, New York: Springer-Verlag, 2003, pp. 39–65.

[30] M. M. S. Daskin, “Network and Discrete Location: Models, Algorithms, and Applications.,” *Wiley, New York*, 1995.

[31] P. S. Davis and T. L. Ray, “A branh-and-bound algorithm for the capacitated facilities location problem,” *Nav. Res. Logist.*, vol. 16, pp. 331–344, 1969.

[32] M. Dorigo and T. Stützle, “The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances,” in *Handbook of Metaheuristics*, Boston: Kluwer Academic Publishers, 2002, pp. 250–285.

[33] Z. Drezner, *Facility location: a survey of applications and methods*. Springer Verlag, 1995.

[34] Z. Drezner and H. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2001.

[35] D. Erlenkotter, “A dual-based procedure for uncapacitated facility location,” *Oper. Res.*, vol. 26, no. 6, pp. 992–1009, 1978.

[36] A. T. Ernst and M. Krishnamoorthy, “Solution algorithms for the capacitated single allocation hub location problem,” *Ann. Oper. Res.*, vol. 86, no. 0, pp. 141–159, 1999.

[37] A. T. Ernst and M. Krishnamoorthy, “Exact and heuristic algorithms for

the uncapacitated multiple allocation p-hub median problem," *Eur. J. Oper. Res.*, vol. 104, no. 1, pp. 100–112, 1998.

[38] A. T. Ernst and M. Krishnamoorthy, "Efficient Algorithms for the Uncapacitated Single Allocation p-Hub Median Problem," *Locat. Sci.*, vol. 4, no. 3, pp. 139–154, 1996.

[39] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Oper. Res.*, vol. 11, no. 3, pp. 399–417, 1963.

[40] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Heidelberg, Germany: Springer-Verlag, 2009.

[41] R. Z. Farahani, M. Hekmatfar, A. B. Arabani, and E. Nikbakhsh, "Hub location problems: A review of models, classification, solution techniques, and applications," *Comput. Ind. Eng.*, vol. 64, no. 4, pp. 1096–1109, 2013.

[42] E. Feldman, F. A. Lehrer, and T. L. Ray, "Warehouse Location Under Continuous Economies of Scale," *Manage. Sci.*, vol. 12, no. 9, pp. 670–684, 1966.

[43] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *J. Glob. Optim.*, vol. 134, pp. 109–134, 1995.

[44] P. Festa and M. G. C. Resende, "GRASP: An annotated bibliography," *Essays Surv. metaheuristics*, pp. 1–41, 2002.

[45] D. Gamboa, "Adaptive Memory Algorithms for the Solution of Large Scale Combinatorial Optimization Problems, PhD Thesis (in Portuguese)," Instituto Superior Técnico, Universidade Técnica de Lisboa, 2008.

[46] D. S. Garey and M. R. Johnson, *Computers and Intractablity: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman & Co. New York, 1979.

[47] S. Gelareh, "Hub Location Models in Public Transport Planning," VDM Verlag, 2008.

[48] M. Gendreau, "An Introduction to Tabu Search," in *Handbook of Metaheuristics*, Boston: Kluwer Academic Publishers, 2003, pp. 37–54.

[49] A. Geoffrion, "Lagrangean relaxation for integer programming,"

Approaches to Integer Program., 1974.

[50] F. Glover, "A template for scatter search and path relinking," *Artif. Evol.*, 1998.

[51] F. Glover, "Tabu search—part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[52] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control Cybern.*, vol. 39, pp. 653--684, 2000.

[53] F. Glover, "Adaptive Memory Projection Methods for Integer Programming," in *Metaheuristic Optimization via Memory and Evolution*, Boston: Kluwer Academic Publishers, 2005, pp. 425–440.

[54] F. Glover, "Tabu search and adaptive memory programming-advances, applications and challenges," *Interfaces Comput. Sci. Oper. Res.*, vol. 1, pp. 1–77, 1996.

[55] F. Glover, "Tabu search—part II," *ORSA J. Comput.*, vol. 2, no. I, pp. 4–32, 1990.

[56] F. Glover, "HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS," *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, Jan. 1977.

[57] F. Glover, "Tutorial on surrogate constraint approaches for optimization in graphs," *J. Heuristics*, vol. 9, no. 3, pp. 175–227, 2003.

[58] F. Glover and M. Laguna, "Tabu Search," 1997.

[59] A. J. Goldman, "Optimal location for centers in a network," *Transp. Sci.*, vol. 3, no. 4, pp. 352–360, 1969.

[60] G. Guastaroba and M. G. Speranza, "Kernel search for the capacitated facility location problem," *J. Heuristics*, vol. 18, no. 6, pp. 877–917, Sep. 2012.

[61] G. Guastaroba and M. G. Speranza, "Kernel Search: An application to the index tracking problem," *Eur. J. Oper. Res.*, vol. 217, no. 1, pp. 54–68, 2012.

[62] M. Guignard, "Lagrangean relaxation," *Top*, vol. 11, no. 2, pp. 151–228, 2003.

[63] M. Guignard and K. Spielberg, "A direct dual method for the mixed plant location problem with some side constraints," *Math. Program.*, vol. 17, no. January 1977, pp. 198–228, 1979.

[64] S. L. Hakimi, “Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph,” *Oper. Res.*, vol. 12, no. 3, pp. 450–459, 1964.

[65] M. Held and R. M. Karp, “The traveling salesman problem and minimum spanning trees - Part II,” *Oper. Res.*, vol. 18, pp. 1138–1162, 1970.

[66] F. Itwm, “Facility Location and Supply Chain Management – A comprehensive review,” *Berichte des Fraunhofer ITWM*, vol. 130, no. 130, p. 63, 2007.

[67] S. K. Jacobsen, “Heuristics for the capacitated plant location model,” *Eur. J. Oper. Res.*, vol. 12, pp. 253–261, 1983.

[68] H. Jia, F. Ordóñez, and M. Dessouky, “A modeling framework for facility location of medical services for large-scale emergencies,” *IIE Trans.*, pp. 1–30, 2007.

[69] R. V. Kennington, Jeff L. and Helgason, *Algorithms for network programming*. John Wiley & Sons, Inc., 1980.

[70] J. G. Klincewicz, “Theory and Methodology: Heuristics for the p-hub location problem,” *Eur. J. Oper. Res.*, vol. 53, pp. 25–37, 1991.

[71] A. Klose and A. Drexl, “Facility location models for distribution system design,” *Eur. J. Oper. Res.*, vol. 162, no. 1, pp. 4–29, 2005.

[72] A. Klose and S. Görtz, “A branch-and-price algorithm for the capacitated facility location problem,” *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 1109–1125, Jun. 2007.

[73] C. Koulamas, S. Antony, and R. Jaen, “A survey of simulated annealing applications to operations research problems,” *Omega*, vol. 22, no. 1. pp. 41–56, 1994.

[74] A. Kuehn and M. Hamburger, “A heuristic program for locating warehouses,” *Manage. Sci.*, vol. 9, pp. 643–666, 1963.

[75] M. Labbé, H. Yaman, and E. Gourdin, “A branch and cut algorithm for hub location problems with single assignment,” *Math. Program.*, vol. 102, no. 2, pp. 371–405, 2005.

[76] C. Lagos, G. Guerrero, E. Cabrera, S. Niklander, F. Johnson, F. Paredes, and J. Vega, “A Matheuristic Approach Combining Local Search and Mathematical Programming,” *Sci. Program.*, vol. 2016, pp. 1–7, 2016.

[77] M. Laguna and R. Martí, *Scatter Search*, vol. 24. Boston, MA: Springer US, 2003.

[78] M.-C. Lai, H. Sohn, T.-L. (Bill) Tseng, and C. Chiang, "A hybrid algorithm for capacitated plant location problem," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8599–8605, Dec. 2010.

[79] L. Lasdon, A. Duarte, F. Glover, M. Laguna, and R. Martí, "Adaptive memory programming for constrained global optimization," *Comput. Oper. Res.*, vol. 37, no. 8, pp. 1500–1509, 2010.

[80] E. Lawler and D. Wood, "Branch-and-bound methods: A survey," *Oper. Res.*, pp. 699–719, 1966.

[81] T. Levanova and E. Tkachuk, "Development of a Bee Colony Optimization Algorithm for the Capacitated Plant Location Problem," in *II International Conference Optimization and Applications (OPTIMA-2011)*, 2011, pp. 153–156.

[82] L. Lorena and E. Senne, "Improving traditional subgradient scheme for Lagrangean relaxation: an application to location problems," *Int. J. Math. Algorithms*, vol. 1, pp. 133–151, 1999.

[83] K. Lu and C. Ting, "Lagrangian Relaxation for the Capacitated Single Allocation p-Hub Median Problem," in *Proceedings of the Eastern Asia Society for Transportation Studies*, 2013, vol. 10, pp. 851–863.

[84] M. Marić, "Variable Neighborhood Search for Solving the Capacitated Single Allocation Hub Location Problem," in *Serdica, Journal of Computing*, 2014, vol. 7, no. 4, pp. 343–354.

[85] S. Martello, D. Pisinger, P. Toth, S. Martello, D. Pisinger, and P. Toth, "New trends in exact algorithms for the 0-1 knapsack problem," *Eur. J. Oper. Res.*, vol. 123, no. 1, pp. 325–332, 2000.

[86] R. Martí, M. Laguna, and F. Glover, "Principles of scatter search," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 359–372, Mar. 2006.

[87] L. Michel and P. Van Hentenryck, "A simple tabu search for warehouse location," *Eur. J. Oper. Res.*, vol. 157, pp. 576–591, 2004.

[88] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.

[89] N. Mohammad, "Using Genetic Algorithms for the Single Allocation Hub

Location Problem," 2009.

[90] S. Nickel, *Location Theory*. Berlin/Heidelberg: Springer-Verlag, 2005.

[91] S. Nickel, Anita Schöbel, and Tim Sonneborn, "Hub location problems in urban traffic networks," *Math. methods Optim. Transp. Syst.*, pp. 95–107, 2001.

[92] M. E. O'Kelly, "The Location of Interacting Hub Facilities," *Transp. Sci.*, vol. 20, no. 2, pp. 92–106, 1986.

[93] M. E. O'Kelly, D. Bryan, D. Skorin-Kapov, and J. Skorin-Kapov, "Hub network design with single and multiple allocation: A computational study," *Locat. Sci.*, vol. 4, no. 3, pp. 125–138, 1996.

[94] M. E. O'Kelly, "A quadratic integer program for the location of interacting hub facilities," *Eur. J. Oper. Res.*, vol. 32, no. 3, pp. 393–404, 1987.

[95] M. E. O'Kelly and H. J. Miller, "The hub network design problem: A review and synthesis," *J. Transp. Geogr.*, vol. 2, no. 1, pp. 31–40, 1994.

[96] M. Paris and C. C. Ribeiro, "Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment," *INFORMS J. Comput.*, vol. 12, no. 3, pp. 164–176, 2000.

[97] M. . Perez, F. Almeida, and J. Marcos Moreno-Vega, "A hybrid GRASP-Path Relinking algorithm for the capacitated p - Hub median problem," *Lecture Notes in Computer Science*, vol. v 3636. pp. 142–153, 2005.

[98] A. Rahmani and S. A. Mirhassani, "A hybrid Firefly-Genetic Algorithm for the capacitated facility location problem," *Inf. Sci. (Ny).*, vol. 283, no. 1, pp. 70–78, 2014.

[99] M. Randall, "Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation," *Comput. Optim. Appl.*, vol. 39, no. 2, pp. 239–261, 2008.

[100] C. Rego, "RAMP: A new metaheuristic framework for combinatorial optimization," in *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, vol. 30, C. Rego and B. Alidaee, Eds. Kluwer Academic Publishers, 2005, pp. 441–460.

[101] C. Rego, F. Mathew, and F. Glover, "RAMP for the capacitated minimum spanning tree problem," *Ann. Oper. Res.*, vol. 181, no. 1, pp. 661–681, Oct. 2010.

[102] C. Rego and L. Sagbansua, “A RAMP Algorithm for the Multi-Resource Generalized Assignment Problem,” University of Mississippi, 2006.

[103] M. G. C. Resende and R. F. Werneck, “A hybrid multistart heuristic for the uncapacitated facility location problem,” *Eur. J. Oper. Res.*, vol. 174, no. 1, pp. 54–68, Oct. 2006.

[104] C. S. ReVelle, H. a. Eiselt, and M. S. Daskin, “A bibliography for some fundamental problem categories in discrete location science,” *Eur. J. Oper. Res.*, vol. 184, no. 3, pp. 817–848, Feb. 2008.

[105] C. S. Revelle and H. A. Eiselt, “Location analysis: A synthesis and survey,” *Eur. J. Oper. Res.*, vol. 165, no. 1, pp. 1–19, 2005.

[106] C. Riley, C. Rego, and H. Li, “A simple dual-RAMP algorithm for resource constraint project scheduling,” in *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE ’10*, 2010, p. 1.

[107] V. F. Ronaldo Silva, “Um Algoritmo GRASP Híbrido para o Problema de Localização Capacitada de Custo Fixo,” Universidade Federal do Rio de Janeiro, COPPE, 2007.

[108] T. Van Roy, “A cross decomposition algorithm for capacitated facility location,” *Eur. J. Oper. Res.*, vol. 34, no. 1, pp. 145–163, 1986.

[109] G. Sa, “Branch-and-Bound and Approximate Solutions to the Capacitated Plant-Location Problem,” *Operations Research*. 1969.

[110] L. Sagbansua, “Algorithms for very large scale multi-resource generalized assignment problems,” University of Mississippi University, MS, USA, 2004.

[111] K. Sastry, D. E. Goldberg, and G. Kendall, “Genetic Algorithms,” in *Search Methodologies*, Boston, MA: Springer US, 2014, pp. 93–117.

[112] R. Sridharan, “A heuristic Lagrangean algorithm for the capacitated plant location problem—A comment,” *Eur. J. Oper. Res.*, vol. 23, no. 2, pp. 264–265, 1986.

[113] R. Sridharan, “The capacitated plant location problem,” *Eur. J. Oper. Res.*, vol. 87, no. 2, pp. 203–213, Dec. 1995.

[114] Z. Stanimirović, “Solving the Capacitated Single Allocation Hub Location Problem Using Genetic Algorithm,” in *Recent Advances in Stochastic Modeling and Data Analysis*, 2007, vol. 1, pp. 464–471.

[115] Z. Stanimirović, “Solving the Capacitated Single Allocation Hub Location

Problem Using Genetic Algorithm," in *Recent Advances in Stochastic Modeling and Data Analysis*, 2007, pp. 464–471.

[116] Z. Stanimirović, "A genetic algorithm approach for the capacitated single allocation p-hub median problem," in *Computing and Informatics*, 2010, vol. 29, no. 1, pp. 117–132.

[117] P. Stanojević, M. Marić, and Z. Stanimirović, "A hybridization of an evolutionary algorithm and a parallel branch and bound for solving the capacitated single allocation hub location problem," *Appl. Soft Comput.*, vol. 33, pp. 24–36, 2015.

[118] J. U. Sun and D. Park, "An Ant Colony System Hybridized with a Genetic Algorithm for the Capacitated Hub Location Problem," pp. 173–181, 2012.

[119] M. Sun, "Solving the uncapacitated facility location problem using tabu search," *Comput. Oper. Res.*, vol. 33, no. 9, pp. 2563–2589, 2006.

[120] M. Sun, "A tabu search heuristic procedure for the capacitated facility location problem," *J. Heuristics*, vol. 18, no. 1, pp. 91–118, 2012.

[121] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Y. Potvin, "Adaptive memory programming: A unified view of metaheuristics," *Eur. J. Oper. Res.*, vol. 135, no. 1, pp. 1–16, 2001.

[122] J. Ting, R. Lu, and C. Wu, "Reactive GRASP for the Capacitated Single Allocation p - Hub Median Problem," *East. Asia Soc. Transp. Stud.*, vol. 11, pp. 819–833, 2015.

[123] H. Venables and A. Moscardini, "Ant based heuristics for the capacitated fixed charge location problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5217 LNCS, pp. 235–242.

[124] V. Verter, *Foundations of Location Analysis*, vol. 155. Boston, MA: Springer US, 2011.

[125] L.-Y. Wu, X.-S. Zhang, and J.-L. Zhang, "Capacitated facility location problem with general setup cost," *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1226–1241, May 2006.